VŠB – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

DIPLOMOVÁ PRÁCE

2014 **Bc. Jozef Marmoľ**

Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Pokročilá komunikační řešení v IP telefonii

Advanced communication solutions in IP telephony

2014 **Bc. Jozef Marmoľ**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Jozef Marmoľ**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2601T013 Telekomunikační technika

Téma: Pokročilá komunikační řešení v IP telefonii
Advanced Communication Solutions in IPTelephony

Zásady pro vypracování:

Cílem diplomové práce je praktická realizace řešení IP telefonie s podporou webRTC a srovnání této perspektivní formy se stávajícími způsoby přenosu jako RTP, SRTP či ZRTP.

1. IP telefonie, protokoly RTP, SRTP, ZRTP a platforma webRTC.
2. Aktuální stav projektu SW PBX Asterisk.
3. Realizace webRTC komunikace na Asterisku s využitím vhodných knihoven.
4. WebRTC z pohledu kvalitativních parametrů a požadavků na síť.
5. Srovnání webRTC komunikace s jinými běžně používanými protokoly v IP telefonii.
6. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

1. M. VOZŇÁK. *Voice over IP*. Ostrava : VŠB TU Ostrava, 2008. 176 s. VŠ skripta. ISBN 978-80-248-1828-3.
2. R. BRYANT, L. MADSEN, J. MEGGELEN. *Asterisk: The Definitive Guide*, 4th Edition, Publisher: O'Reilly Media, May 2013, p. 846.
3. M. VOZNAK, K. TOMALA, J. VYCHODIL, J. SLACHTA. *Advanced concept of voice communication server on embedded platform.* Przeglad Elektrotechniczny, Volume 89, Issue 2 B, 2013, pp. 228-233.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Miroslav Vozňák, Ph.D.**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2014

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlásenie študenta

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave dňa: *6. mája 2014*

.....
podpis študenta

Poďakovanie

Rád by som sa poďakoval doc. Ing. Miroslavovi Vozňákovi, Ph.D. za odbornú pomoc a konzultáciu pri vytváraní tejto diplomovej práce.

Abstrakt

Práca sa zaoberá tematikou pokročilej komunikačnej služby, ktorá môže byť v modernej pobočkovej ústredni ponúkaná. Je zameraná hlavne na otvorené a voľne šíriteľné aplikácie. Dôraz je kladený na technológiu WebRTC a sipML5 klienta v spolupráci s pobočkovou ústredňou Asterisk. Praktická časť sa venuje konfigurácii pobočkovej ústredne Asterisk tak aby pracovala so sipML5 klientom a následným analyzovaním spojenia so stávajúcim riešením. Obsahuje detailný rozbor spojenia či už v spojení s doterajším bežne používaným SIP klientom alebo ďalším sipML5 klientom cez webové rozhranie. V práci je vykonaná aj analýza spojenia, na ktorú som použil paketový analyzátor Wireshark. Vďaka podrobnej dokumentácii o konfigurácii virtuálneho stroja je možné ľahko projekt zrekonštruovať a nadviazať na neho v ďalších projektoch a prácach.

Kľúčové slová

Asterisk; RTP; SRTP; ZRTP; sipML5; WebRTC; webrtc2sip; websocket

Abstract

The work deals with the topic of advanced communication service that can be offered in a modern private branch exchange. Mainly it is focused on open and free distributable applications. The emphasis is placed on technology WebRTC and sipML5 of client in cooperation with private branch exchange Asterisk. The practical part describes the configuration of private branch exchange Asterisk to work with sipML5 client and then analyzing the connections with an existing (building) solution. It contains a detailed analysis of the connection either in connection with the present commonly used SIP client or another sipML5 client through the web interface. In this work is also performed analysis of the connection, for which I used a packet analyzer Wireshark. Thanks to the detailed documentation about the configuration of the virtual machine it is possible to reconstruct project easily and to build on it in other projects and works.

Key words

Asterisk; RTP; SRTP; ZRTP; sipML5; WebRTC; webrtc2sip; websocket

Zoznam použitých skratiek

Skratka	Význam
IETF	Internet Engineering Task Force
QoS	Quality of Service
STUN	Session Traversal Utilities NAT
SRTP	Secure Real-time Transport Protocol
RTCP	Real Time Control Protocol
RTP	Real Time Protocol
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SRTCP	Secure Real-time Transport Control Protocol
ZRTP	Zimmermann Real Time Protocol

Obsah

Úvod

Pobočkové ústredne sa stali dôležitou súčasťou každej väčšej spoločnosti v oblasti komunikácie. S postupom času sa stali z obyčajných prepojovacích zariadení modernejšie komunikačné centrá. A tak aj s rozvojom integrácie rôznych hlasových služieb do pobočkovej ústredne Asterisk sa rozvíjajú aj technológie umožňujúce samotné spojenie klientskych zariadení. Tieto nové technológie kladú veľký dôraz na skvalitnenie spojenia a v neposlednom rade hlavne na bezpečnosť.

Možnosť komunikácie s využitím „desktopových“ SIP klientov sa stala bežnou súčasťou a práve preto sa vývoj snaží čo najviac uľahčiť prácu s týmito SIP klientskymi aplikáciami. Užívatelia sa stávajú stále viac a viac náročnejšími v zmysle komfortnosti a snažia sa vyhýbať rôznym inštaláciám alebo zložitým kompiláciám. Aplikácia sipML5 chce zbaviť koncového užívateľa starosti o samotnú inštaláciu a možnosti jej zložitej konfigurácie. Ďalším faktorom, ktorého sa chcú bežný užívatelia zbaviť je rôznorodosť. Užívatelia sa neradi neustále zoznamujú s novým prostredím aplikácií. SipML5 ponúka jednotné rozhranie na akomkoľvek zariadení bez nutnosti inštalácie. A keďže každé klientske zariadenie využíva webový prehliadač, tak je výhodné využiť pravé toto rozhranie ako samotný komunikačný nástroj.

Technológiu WebRTC využívajú aj komunikačné prostriedky ako appear.in alebo imo.im. No stále sa jedná o ďalšiu stranu cez ktorú pôjde dátový tok a nemáme tak úplnú kontrolu nad dátovým tokom. Moja diplomová práca nevyužíva takúto ďalšiu stranu a je čisto vytváraná na mnou (myslí sa vo vlastníctve správcu softvérovej ústredne Asterisk) spravovaných aplikáciách a zariadeniach.

Diplomová práca sa zaoberá problematikou pokročilých služieb v modernej pobočkovej ústredni. Praktická časť je zameraná na voľne šíriteľné aplikácie. Základným podstavcom pre túto prácu bude pobočková ústredňa Asterisk.

Detailne sa tu budem venovať technológii zvanej WebRTC a aplikácii, ktorá túto technológiu využíva aplikácii sipML5. Celá podpora technológie WebRTC a aplikácie sipML5 je spustená práve na pobočkovej ústredni Asterisk verzie 11. Ďalšou voľne šíriteľnou aplikáciou, ktorá je použitá je webový prehliadač Google Chrome. Dôvodom prečo práve Chrome je ten, že ako jediný zatiaľ ponúka plnú podporu technológii WebRTC bez potreby doinštalovania si akejkoľvek aplikácie ako je to u iných webových prehliadačov Firefox Mozilla, Opera alebo Safari, kde je potrebná inštalácia balíčka webrtc4all.

Cieľom mojej diplomovej práce má byť praktická realizácia riešenia využívajúca WebRTC a jeho porovnanie s riešením založeným na protokole SIP.

1 Protokoly

1.1 H.323

H.323 je prvým protokolom. Protokol H.323 je schopný zaistiť konferenčné hovory obsahujúce hlasový, obrazový a dátový prenos. Samotný Asterisk neobsahuje podporu H.323 protokolu. K tomu slúži modul asterisk-oh323. Nástupcom H.323 protokolu je v súčasnosti protokol SIP. [14, 21]

1.2 SIP (Session Initiation Protocol)

SIP je signalizačný protokol pre VOIP hovory. SIP na rozdiel od H.323 je už štandardnou súčasťou inštalácie Asterisku.

Z hľadiska syntaxe je veľmi podobný protokolom HTTP alebo SMTP. SIP bol pôvodne odoslaný do IETF (Internet Engineering Task Force) v roku 1996. Trvalo 3 roky než bol po 11 revíziach uvedený SIP RFC 2543.

SIP je protokol aplikačnej vrstvy. Ku komunikácii využíva TCP aj UDP protokol transportnej vrstvy a port 5060. Slúži k vybudovaniu, modifikácii a ukončeniu multimediálnych relácií (napr.: telefónne hovory cez internet).

K prenosu médií medzi koncovými bodmi využíva SIP protokol RTP (Real Time Protocol). RTP využíva k prenosu médií vysoké čísla portov a to od 10 000 – 20 000.

1.3 RTP (Real-time Transport Protocol)

Ako som už spomínal RTP sa využíva k prenosu médií medzi koncovými bodmi. Za média sa tu považujú aplikácie, ktoré vyžadujú prenos dát v reálnom čase, ako sú audio, video alebo simulácie dát cez multikástové ale unikástové sieťové služby. Tieto služby sú zahrnuté v type identifikátora, sekvenčného čísla, značkovania a monitorovaniu doručenia. Aplikácie RTP zväčša bežia na vrchole UDP a využívajú tomu služby multiplexovania a kontrolného súčtu. RTP podporuje prenos dát k viacerých cieľom s využitím distribúcie multikástu. [9]

RTP si nerezervuje adresovanie zdrojov a nezaručuje kvalitu služby (QoS) pre služby idúce v reálnom čase. Samotné RTP neposkytuje žiadny mechanizmus k zabezpečeniu včasného doručenia alebo garancie poskytnutia iných QoS služieb. Avšak RTP sa spolieha na služby nižších vrstiev. To znamená, že nezaručuje doručenie alebo prevenciu doručenia služieb mimo prevádzky, ani nepredpokladá, že samotná základná sieť zodpovedá a doručí pakety v správnom poradí. Sekvenčné číslo zahrnuté v RTP povoľuje prijímateľovi znovu usporiadať pakety v správnom poradí ako boli odoslané odosielateľom. Zároveň môže byť sekvenčné číslo použité k zisteniu alebo vymedzenia správneho umiestnenia paketu. Používa sa to napríklad k dekódovaniu videa.

Prenos dát je len rozšírením riadiaceho protokolu (RTCP) umožňujúceho sledovanie doručenia dát vo veľkých multikástových sieťach a zároveň zabezpečuje minimálnu kontrolu a identifikáciu funkčnosti.

RTP a RTCP sú vytvorené nezávisle na podliehaní transportnej a sieťovej vrstve.

1.3.1 **Definovanie pojmov v RTP**

RTP payload (RTP zaťaženie)- dáta prenášané RTP v pakete. Napríklad audio vzorky alebo stlačeného (compressed) videa.

RTP paket- paket pozostáva z RTP hlavičky čo predstavuje pevnú časť paketu a z dát payload-u. Niektoré základné protokoly môžu vyžadovať enkapsuláciu RTP protokolu. Typicky jeden paket zo základných protokolov obsahuje čistý RTP paket, ale niekoľko RTP paketov môžu byť obsiahnuté, ak je povolená, enkapsulačnou metódou.

RTCP paket- kontrolný paket pozostáva z pevnej časti čo je hlavička. Podobá sa RTP hlavičke. Táto hlavička je ďalej nasledovaná prvkami, ktoré sa líšia v závislosti na type RTCP paketu. Typicky viaceré pakety sú posielané spoločne ako nejaká zlúčenina RTCP paketu v jednom pakete zo základných protokolov. Takéto posielanie je automaticky spúšťané dĺžkou poľa v pevnej časti RTCP paketu (hlavičke).

Port- je „abstrakt, ktorý transportné protokoly požívajú k rozlišovaniu medzi viacerými cieľmi v rámci jedného počítača (hosta). TCP/IP protokoly identifikujú porty použitím malého kladného čísla“. Transportné selektory (TSEL- transport selectors) používané OSI transportnou vrstvou sú rovnocenné k týmto portom. RTP závisí na protokole nižšej vrstvy k poskytnutiu nejakého mechanizmu, ako ked sú napríklad porty multiplexované na RTP a RTCP pakety v relácii.

Transportná adresa- je kombinácia sieťovej adresy a portu, ktorý identifikuje úroveň prenosu koncového bodu. Napríklad IP adresa a UDP port. Pakety sú prenášané zo zdrojovej adresy do cieľovej adresy.

RTP relácia (session)- spojenie medzi skupinou účastníkov komunikujúcich s RTP. Pre každého účastníka, relácia je definovaná príslušným párom cieľových adries. Cieľová adresa môže byť spoločná pre všetkých účastníkov (v prípade multicastu) alebo môže byť rozličná pre každého (v prípade unicastu plus spoločný pár portu). V multimediálnej relácii, každé médium je nesené v inej RTP relácii s jeho vlastnými RTCP paketmi. Viaceré RTP relácie sú rozdelené rozdielnym číslom páru portu a alebo rozdielnymi multicastovými adresami.

Zdroj synchronizácie (SSRC- Synchronization source)- je zdroj toku RTP paketov. Sú identifikované 32-bitovým numerickým SSRC ukazovateľom, ktorý je nesený v RTP hlavičke takže nie je závislý na adrese siete.

Prispievajúci zdroj (CSRC- Contributing source)- zdroj toku RTP paketov, ktoré prispievajú ku kombinovanému toku vyprodukovanému z RTP mixéru. Mixér vkladá zoznam

SSRC ukazovateľov zdrojov, ktorí prispievajú do tvorenia konkrétneho paketu v RTP hlavičke istého paketu. Tento zoznam sa nazýva CSRC zoznam. Príkladom môže byť audio konferencia kde mixér uvádza všetkých hovoriacich, ktorí reč bola zložená a tak sa vytvoril odchádzajúci paket s tým, že prijímač povolil uvedenie tohto hovoriaceho aj keď všetky audio pakety obsahujú rovnaký SSRC ukazovateľ.

Koniec systému (End system)- je aplikácia ktorá vytvára obsah, ktorý ma byť posielaný v RTP paketoch a/alebo „konzumuje“ obsah prijatých RTP paketov. Môže slúžiť ako jeden alebo viacero zdrojov synchronizácie v určitej RTP relácii, avšak je to typický len jeden.

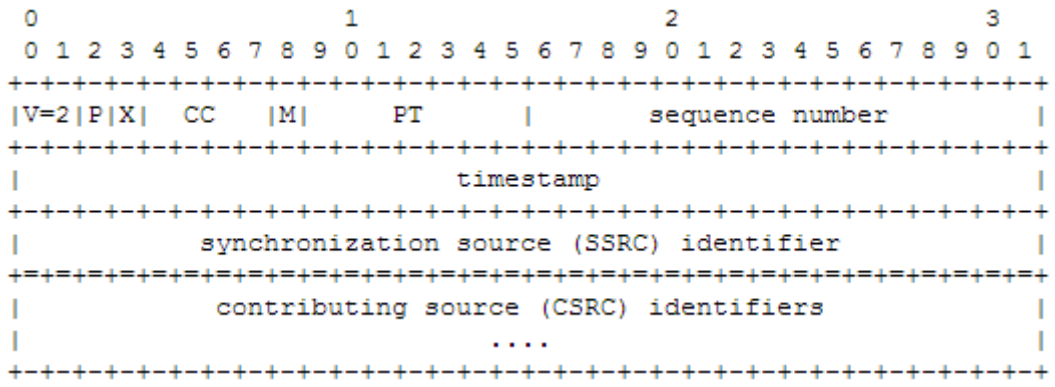
Mixér (Mixer)- je „medziľahlý“ systém, ktorý prijíma RTP pakety z jedného alebo z viacerých zdrojov. Môže meniť formát dát, kombinovať pakety a potom ďalej odovzdáva nový RTP paket.

Prekladač (Translator)- je „medziľahlý“ systém, ktorý posiela RTP pakety s ich neporušeným identifikátorom zdroja synchronizácie.

Monitor- aplikácia, ktorá prijíma RTCP pakety posielané účastníkmi v RTP relácii, najmä príjem správ a odhaduje súčasnú kvalitu služby (QoS) pre distribučné monitorovanie, diagnostiku chýb a dlhodobé štatistiky

1.3.2 Hlavička RTP protokolu

Hlavička RTP vyzerá nasledovne:



Obrázok 1.1: *Formát RTP hlavičky*

Prvých 12 oktetov sa nachádza v každom RTP pakete, zatiaľ čo zoznam CSRC identifikátorov sa nachádza v RTP pakete až keď sú vložené mixérom. Popis hlavičky RTP:

version (V): 2 bity – (verzia) toto pole identifikuje verziu RTP.

padding (P): 1 bit – (vypchanie) ak je tento bit nastavený, tak paket obsahuje jednu alebo viac prídavných padding oktetov na konci, ktoré nie sú súčasťou zaťaženia (payload). Posledný oktet padding-u obsahuje počet, koľko padding oktetov by malo byť ignorovaných.

extension (X): 1 bit – (predĺženie) ak je tento bit nastavený, tak pevná časť hlavičky je nasledovaná presne jednou hlavičkou predĺženia s presne definovaným formátom.

Marker (M): 1 bit – (značka) výklad značky je definovaným profilom. Je určená k povoleniu určitých veci ako je napríklad, že hranica rámca je označená v toku paketov.

Payload type (PT): 7 bitov – (typ zaťaženia) toto pole identifikuje formát RTP zaťaženia (payload-u) a podáva jej výklad aplikácii.

Sequence number: 16 bitov – (sekvenčné číslo) sekvenčné číslo narastá po jednom pre každý odoslaný dátový RTP paket. Môže byť využívané prijímateľom k odhaleniu stratovosti paketu a k obnoveniu postupnosti paketov.

Timestamp: 32 bitov – (časová značka) zobrazuje čas (okamih) vzorkovania prvého oktetu v dátovom RTP pakete. Vzorkovací čas (okamih) musí byť odvodený od hodín, ktoré zaisťujú lineárne zvyšovanie času k umožneniu synchronizácie a prepočtu veľkosti oneskorenia paketov (jitter).

SSRC: 32 bitov – toto pole identifikuje zdroj synchronizácie. Samotný identifikátor je vybraný náhodne s tým, že žiadne dva synchronizačné zdroje v rámci rovnakej RTP relácie nemajú rovnaký identifikátor.

CSRC list: 0 až 15 zložiek, 32 bitov pre každú zložku – (CSRC zoznam) CSRC zoznam označuje (identifikuje) zdroje „prispievajúceho“ do užitočného zaťaženia (payload) obsiahnutého v tomto pakete. Množstvo identifikátorov je daných CC poľom. Ak je tam viac ako 15 prispievajúcich zdrojov potom len 15 môže byť určených. CSRC identifikátory sú vkladané mixérom, s pomocou SSRC identifikátorov z prispievajúcich zdrojov.

1.4 RTCP (RTP Control Protocol)

Tento protokol je založený na periodickom prenose kontrolných paketov každému účastníkovi v relácii, využitím rovnakých distribučných mechanizmov ako dátových paketov.

Základný protokol (underlying protocol) musí poskytovať multiplexovanie dátových a kontrolných paketov. Napríklad použitie oddelenie čísla portu s UDP.

1.4.1 Typy správ RTCP

Špecifikácia niekoľkých RTCP správ v závislosti na type prenášanej kontrolnej informácie.

SR (Sender Report) je určený pre prenos a príjem štatistík od účastníkov, ktorý sú aktívnymi odosielateľmi. Správy tohto typu obsahujú informácie o aktuálne prebiehajúcej komunikácii. Sender Report zahŕňa v sebe časové značky (timestamps), v ktorých je uložené číslo v sekundách.

RR (Receiver Report) je určený pre prijímanie štatistík od účastníkov, ktorý nie sú aktívnymi odosielateľmi. Informujú odosielateľov a príjemcov o kvalite služieb. Obsahujú čísla stratených paketov a tiež informácie o kolísaní oneskorenia (jitter) pri prijímači.

SDES (Source Description items) ich pravidelné odosielanie vysielačom informuje ostatných účastníkov o sebe. Môžu byť napríklad CNAME (kanonické meno), NAME (meno úžívateľa), EMAIL (emailová adresa), PHONE (telefónne číslo) a mnoho ďalších.

BYE indikuje koniec spojenia (účasti).

APP slúži zasielaniu správ, ktoré nie sú definované v štandarde, teda umožňuje definíciu nových typov správ. Využívané hlavne pre experimentálne účely.

1.5 SRTP (Secure Real-time Transport Protocol)

Secure Real-time Transpor Protocol je definovaný ako rozšírenie RTP protokolu hlavne v oblasti bezpečnosti a overovania integrity. Konceptuálne sa SRTP dá brať ako implementácia typu „naraziť do zásobníku“ (bump in the stack), ktoré je umiestnená medzi RTP aplikáciou a transportnou vrstvou. SRTP zachytáva RTP pakety a následne ich odosiela ako SRTP pakety. Následne SRTP zachytáva SRTP pakety a prepúšťa ich ako ekvivalent alebo rovnocenné RTP paketu v zásobníku na prijímacej strane. [10]

1.5.1 Formát SRTP paketu

Formát SRTP paketu je zobrazený v prílohe A.1. Naproti RTP paketu tu vidíme na viac dve polia – MKI a Authentication tag.

MKI (Master Key Identifier): jeho dĺžka je konfigurovateľná. Pole MKI je nepovinné (Optional) a identifikuje master key. Od tohto master key sú odvodené tajné symetrické kľúče tzv. kľúče relácie (session keys). Kľúče relácie sú dohodnuté medzi užívateľmi ihneď po naviazaní spojenia a po zvyšok celej relácie sa nimi šifrujú prenášané dáta. Najskôr si však komunikujúce strany musia vymeniť master key. Tento master key im slúži na vygenerovanie všetkých potrebných kľúčov relácie.

Authentication tag: jeho dĺžka je taktiež ako u MKI konfigurovateľná. Na rozdiel od MKI sa authentication tag doporučuje pretože chráni pakety od neautorizovanej zmeny obsahu. Autentizačná časť (authenticated portion) pozostáva z RTP hlavičky za ktorou nasleduje šifrovaná časť (Encrypted portion) SRTP paketu. Teda ak je povolené šifrovanie a autentizácia, tak na strane odosielateľ musí byť šifrovanie povolené pred autentizáciou a naopak na strane prijímateľa.

1.5.2 SRTP kryptografický kontext

Každý SRTP tok potrebuje odosielateľa a prijímateľa k zachovaniu informácie kryptografického stavu. Táto informácia sa nazýva „kryptografický kontext“.

SRTP využíva dva typy kľúčov a to session keys (kľúče relácie) a master keys. „Session“ kľúčom sa myslí kľúč, ktorý je používaný priamo ku kryptografickej premene (ako je šifrovanie alebo autentizácia), a „master“ kľúčom sa myslí náhodný reťazec bitov z ktorého sú odvodené session kľúče kryptografickým bezpečnostným spôsobom.

Master kľúče a ostatné parametre, ktoré sa nachádzajú v kryptografickom kontexte sú poskytované mechanizmom kľúčového manažmentu (key management mechanisms) externe k SRTP.

1.6 SRTCP (Secure Real Time Control Protocol)

SRTCP poskytuje rovnaké bezpečnostné služby RTCP ako ponúka SRTP RTP. Správa SRTCP je povinná čím chráni jednotlivé polia RTCP paketu v udržiavaní vzťahov.

Definícia Secure RTCP je podobná definícii Secure RTP. SRTCP je ale obohatené o ďalšie tri nové polia, ktoré sú povinné (SRTCP index, encrypt-flag, authentication tag) a o jedno voliteľné pole (MKI).

1.6.1 Formát SRTCP paketu

Ukážku SRTCP paketu môžeme vidieť v prílohe A.2.

Šifrovaná časť (Encrypted Portion) SRTCP paketu pozostáva zo zašifrovaného RTCP payload-u (užitočnej záťaže), ktorá je ekvivalentom zlúčeniny RTCP paketu z pôvodného RTCP paketu, to jest od deviateho octetu až po koniec samotnej zlúčeniny RTCP paketu. Autentizačná časť (Authentication Portion) SRTCP paketu pozostáva zo samotnej zlúčeniny RTCP paketu, E-flag a SRTCP index.

E-flag: 1 bit, nutné (required) poukazuje na to či je SRTCP paket zašifrovaný alebo nie. Povoľuje rozdeliť zmiešaný RTCP paket do dvoch paketov z nižších vrstiev (lower-layer). Jeden z nich bude zašifrovaný a druhý bude posielaný ako nezašifrovaný. E bit nastavený na „1“ označuje zašifrovaný paket a „0“ označuje nezašifrovaný paket.

SRTCP index: 31 bitov, nutné (required) je 32 bitové počítadlo pre SRTCP paket. Je zahrnutý v každom pakete. SRTCP index musí byť nastavený na nulu predtým než je odoslaný prvý SRTCP paket. Musí sa zväčšovať po jednom, modulo 2^31, po každom odoslanom SRTCP pakete. Obzvlášť, po znovu vytvorení kľúča nesmie byť SRTCP index zresetovaný začínať od nuly.

Authentication Tag: konfigurovateľná dĺžka, nutné (required) využíva sa k prenášaniu správy autentizačných dát.

MKI: konfigurovateľná dĺžka, voliteľné (optional) je Master Key Indicator.

1.7 ZRTP (Zimmermann RTP)

ZRTP je kryptografický protokol, založený na dohode a výmene kľúčov pre šifrovanie medzi dvoma koncovými bodmi v VoIP (Voice over IP) hovoroch. ZRTP je definovaný ako protokol pre dohadovanie výmeny kľúčov relácie (spojenia) metódou Diffie-Hellman pri nadväzovaní hovoru v reálnom čase (RTP), pričom bol nadväzovaný pomocou iného signalizačného protokolu (SIP). Na základe toho sa vytvorí zdieľaný tajný kľúč pomocou ktorého sa ďalej generujú kľúče pre zabezpečenú RTP (SRTP) komunikáciu. [3, 11]

Protokol ZRTP nepreberá verejný kľúč infraštruktúry (PKI) alebo nevyžaduje zložitosť certifikátov v koncových zariadeniach. Pre masmediálne relácie (spojenia) protokol ZRTP poskytuje dôveryhodnosť, ochranu proti útokom „man-in-the-middle“ (MiTM) a v prípadoch kde signalizačný protokol poskytuje ochranu integrity, autentizáciu.

Tento protokol vie zužitkovať atribúty „Protokolu opisu spojenia“ (SDP- Session Description Protocol) k poskytnutiu zistenia a autentizácie cez signalizačný kanál. Poskytuje QoS typu „best effort“ SRTP, ZRTP využíva RTP/AVP profily. Protokol zabezpečuje masmediálne relácie (spojenia), ktoré zahŕňajú zvukový prenos a taktiež zabezpečuje masmediálne relácie (spojenia), ktoré neobsahujú zvuk použitím ľubovoľných digitálnych podpisov.

ZRTP musí byť multiplexovaný on rovnakých portoch ako pakety RTP média. Používa jedinečnú hlavičku, ktorá je jasné odlíšiteľná od ostatných RTP alebo od nástrojov pre priechod relácií k NAT (STUN- Session Traversal Utilities NAT).

Protokol ZRTP podporuje „objavovanie“ SDP atribútu v signalizačnej ceste s prítomnosťou ZRTP. Avšak v niektorých prípadoch ak nie je prebratý v signalizácii, koncový bod môže zaslať ZRTP Hello správy k tomu aby zistil, či už nie je prijatý. Môže sa stať, že odpoveď nebola prijatá, potom žiadne ďalšie ZRTP správy už nie sú posielané počas relácie (spojenia). Toto sa vykonáva z hľadiska bezpečnosti.

Oba koncové body ZRTP začínajú výmenu ZRTP posielaním ZRTP Hello správy medzi sebou. Účelom tohto je si potvrdiť, že oba konce podporujú protokol a vidia či majú koncové body rovnaké algoritmy.

Hello správa obsahuje konfiguračné vlastnosti SRTP a ZID. Každá inštancia ZRTP má jedinečné náhodné 96 bitové ZRTP ID alebo inak nazývané aj ZID. ZID je vygenerované raz počas inštalácie. ZID je odhaľované počas výmeny Hello správ. Prijaté ZID je využívané k vzhliadnutiu uchovaných zdieľaných informácií z predchádzajúcich ZRTP relácií.

Odpoveďou na ZRTP Hello správu je ZRTP HelloACK správa. Správa HelloACK jednoducho potvrdzuje prijatie Hello správy. Keďže RTP využíva best effort UDP prenos, ZRTP má časovače opakovaného prenosu v prípade stratených datagramov. Nachádzajú sa tam dva časovače. Oba s exponenciálnym spätným mechanizmom. Jeden časovač je využívaný k opakovanému prenosu Hello správ a druhý je využívaný k opakovanému prenosu všetkých ostatných správ po prijatí správy HelloACK.

Hello a ostatné ZRTP správy obsahujú hash, ktorý je používaný k spájaniu správ dohromady.

1.7.1 Vytvorenie SRTP spojenia pomocou ZRTP

Prostriedkom pre výmenu kľúča na založenie SRTP spojenia sa využíva dátová hlasová zložka a nie ako v mnohých iných prípadoch signalizačné pakety (SIP). Tento protokol využíva Diffie-Helman kľúče, ktoré sú novo generované pre každé spojenie pričom sa nevyužíva tretej strany PKI. Avšak samotná výmena kľúča nezabezpečuje ochranu proti MiTM útoku. A práve z tohto dôvodu ZRTP umožňuje dokazovať pravosť kľúča pomocou tzv. krátkej overovacej frázy (SAS- Short Authentication String).

SAS je hash dvoch Diffie-Helman hodnôt. Je zobrazená na oboch koncových stranách, prečítaná a potvrdená druhou stranou.

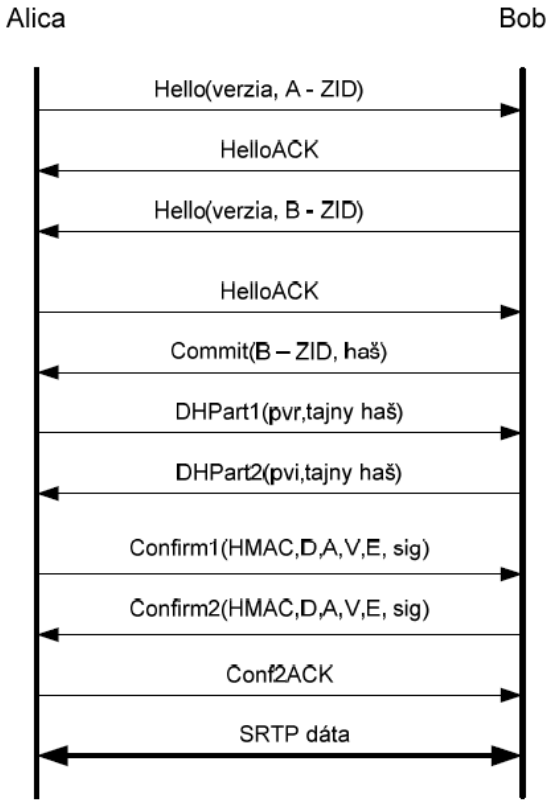
Príklad toku správ u ZRTP je na obrázku 3. Upozorňujem, že Hello/HelloACK môžu ísť aj opačne ako je na obrázku 3. To znamená že čí už Alica alebo Bob môžu poslať správu Hello ako prvý. Je dôležité, že koncový bod, ktorý posielal zaväzujúcu správu (Commit message) je považovaný za iniciátora ZRTP relácie a riadi proces výmeny dohadovania kľúča.

Samotné spojenie sa inicializuje správou Hello. Potvrdzuje sa tým podpora ZRTP protokolu. Hello správa obsahuje aj spomínané ZID číslo.

Druhá strana odpovedá potvrdzujúcou správou HelloACK. ZRTP správy obsahujú hash obraz, ktorým sa spájajú a tým zamedzujú prijímaniu falošných ZRTP správam. Nasleduje zaväzujúca správa (Commit). Zaväzujúca správa charakterizuje použitý kľúčový mód. ZRTP podporuje tri módy.

1.7.2 Diffie-Helmann mód

Koncové strany si vymenia verejné Diffie-Helman hodnoty v správach DHPart1 a DHPart2. Na každej strane sa potom vypočíta SRTP šifrovacie a salt kľúče. Overovací string (SAS) môže byť voliteľne zabezpečený digitálnym podpisov v podobe hodnoty sig v správach Confirm1 a Confirm2 ako aj samotné potvrdenie úspešného výpočtu kľúčov. Confirm správy môžu obsahovať viacero značiek tzv. flags.



Obrázok 1.2: *Vytvorenie SRTP relácie pomocou ZRTP*

1.7.3 **Pred zdieľaný mód (Preshared mode)**

V tomto móde koncové body môžu preskočiť Diffie-Helmanov prepočet pokiaľ majú zdieľaný tajný kľúč z predchádzajúceho ZRTP spojenia (relácie). Tento mód je vyjadrený v zaväzujúcej správe (Commit) a následkom toho je rovnaký volaný tok ako v Multistream móde. Základným rozdielom medzi Multistream a pred zdieľaným módom je, že pred zdieľaný mód používa predchádzajúci zdieľaný tajný kľúč uložený vo vyrovnávajúcej pamäti namiesto aktívneho kľúča ZRTP spojenia.

Tento mód je užitočný pre koncové body, ktoré majú slabší procesor a tým pádom nemusí byť Diffie-Helmanov prepočet spúšťaný pre každé spojenie.

1.7.4 **Multistream mód (Viac tokový mód)**

Tento mód sa využíva keď dva koncové body majú ustálený tok médií SRTP medzi sebou s aktívnym kľúčom ZRTP spojenia. ZRTP dokáže odvodiť viacero SRTP kľúčov z jednej výmeny výpočtu Diffie-Helmana. Napríklad ak ustálený bezpečný „hlasový“ hovor ku ktorému sa pridá ďalší ale video hovor tak v tom prípade sa použije Multistream mód k rýchlemu nadviazaniu video spojenia bez ďalšej výmeny výpočtu Diffie-Helmana.

1.7.5 Vynálezcovia

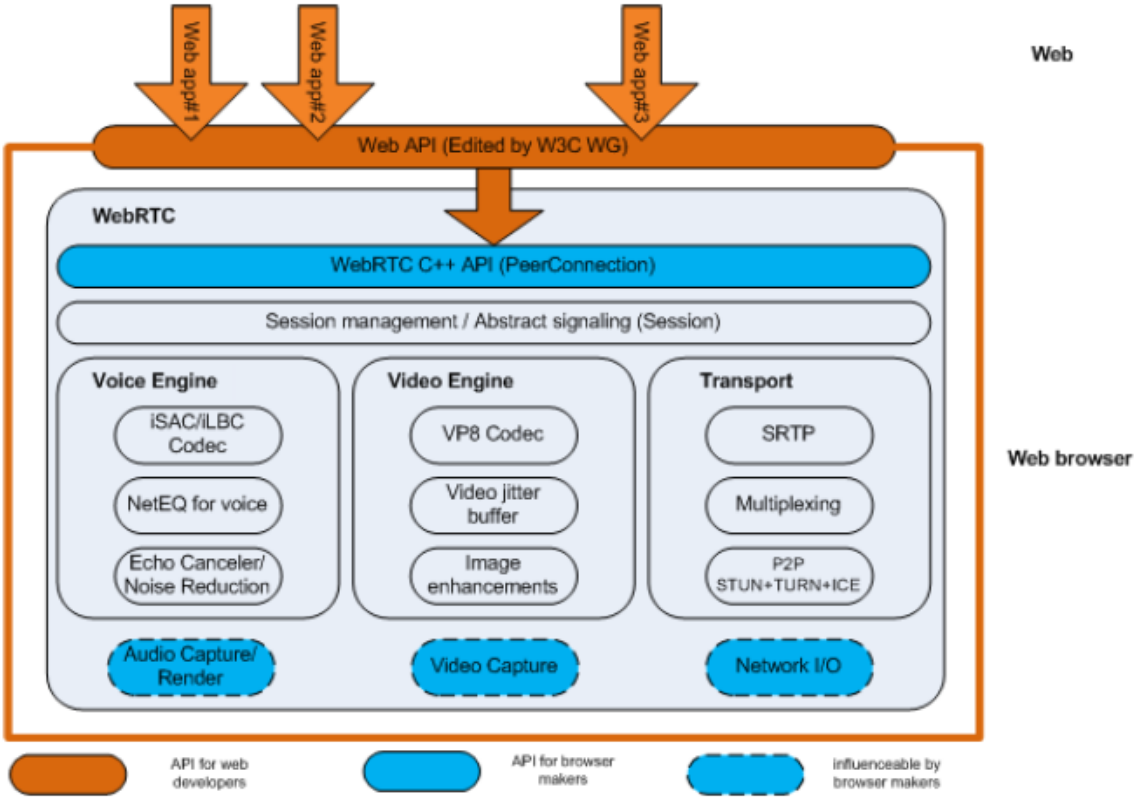
Prvé „vypustenie“ špecifikácie ZRTP protokolu bolo vyvolané a publikované v roku 2006 skupinou expertov v šifrovaní vedenou Philipom Zimmermannom.

1.8 WebRTC

WebRTC je voľne šíriteľný a otvorený projekt, ktorý uspôsobuje spôsobilosť webového prehliadača s komunikáciou v reálnom čase (RTC) pomocou jednoduchých Javaskript aplikácií (API). Hlavnou úlohou projektu je dosiahnutie bohatých a veľmi kvalitných RTC aplikácií. Tieto aplikácie budú vytvorené jednoduchým javaskiptovým API podľa špecifikácie HTML 5. [21]

Mnoho ľudí sa pozerá na WebRTC ako možnosť jeho spojenia v mobilných telefónoch no WebRTC umožní omnoho viac. Jeho aktuálny stav je stále v čase vývoja ale už ho podporuje rada webových prehliadačov (Google Chrome, Firefox a Opera). WebRTC umožňuje vývojárom webových aplikácií schopnosť písať multimediálne aplikácie v reálnom čase na webe bez nutnosti inštalácie akýchkoľvek pluginov.

V súčasnosti podporuje WebRTC Asterisk 11 s tým, že niektoré API sa neustále behom ich vývoja menia. Architektúra je zobrazená na obrázku 1.3.



Obrázok 1.3: *Schéma postupu vývoja [21]*

1.8.1 Web App

Vývojári webových aplikácií (tretia strana vývojárov) s možnosťou AV hovorov a chatov, ktorý sú vyzbrojení, vďaka webRTC, webovým rozhraním API pre túto komunikáciu v reálnom čase.

1.8.2 Web API

API, ktoré má byť použité vývojármi tretích strán pre vývoj webových multimediálnych aplikácií.

1.8.3 WebRTC C++ API

Vrstva API, ktorá umožňuje tvorcom prehliadačov ľahko implementovať návrh Web API.

1.8.4 Transport/Session

Komponenty prenosu sú zostavené z komponenov libjingle. Libjingle je zbierka kódov v C++ a taktiež ukážkových aplikácií, ktoré umožňujú vytvorenie peer-to-peer aplikácií. Kód slúži k vytvoreniu sieťového spojenia (cez NAT, firewally, relay servery a proxy servery).

RTP Stack - jedná sa o sieťový zásobník pre Real Time Protocol.

STUN/ICE - Komponenta umožňujúca používať STUN a ICE mechanizmy k naviazaniu spojenia cez rôzne typy sietí. STUN je zas zbierka pomocných internetových štandardov, ktoré slúžia k umožneniu komunikácie cez NAT, podobne ako ICE.

Session Managment - abstraktná vrstva relácie, ktorá umožňuje nastavenie hovoru a spravovanie vrstvy. Toto necháva rozhodnutie ohľadom implementácie protokolu na vývojároch aplikácií.

1.8.5 VoiceEngine

VoiceEngine je framework pre prenos audia zo zvukovej karty do siete. Framework je softwarová štruktúra, ktorá slúži ako podpora pri programovaní a vývoji aplikácií (knižnice API, navrhované vzory, doporučené postupy).

iSAC - je širokopásmový a super širokopásmový kódek pre VoIP a streamovanie audia. iSAC používa 16 kHz alebo 32 kHz vzorkovaciu frekvenciu s adaptívnou a variabilnou prenosovou rýchlosťou 12 až 52 kbps.

iLBC - jedná sa o úzkopásmový hlasový kódek pre VoIP a streamovanie audia. Používa 8 kHz vzorkovaciu frekvenciu s dátovým tokom 15,2 kbps pre 20 ms rámce a 13,33 kbps pre 30 ms rámce. Je definovaný IETF RFC 3951 a 3952.

Opus - podporuje konštantný i variabilný dátový tok pre kódovanie od 6 kbit/s do 510 kbit/s, pri veľkosti rámca od 2,5 ms do 60 ms, a rôzne vzorkovacie rozsahy od 8 kHz (s šírkou pásma 4 kHz) až do 48 kHz (s 20 kHz šírkou pásma, kde je obsiahnutý rozsah počuteľnosti ľudského sluchového systému). Je definovaný v IETF RFC 6176.

1.8.6 NetEQ for Voice

Dynamický jitter buffer a algoritmus odstraňovania chýb používaný pre skrývanie negatívnych účinkov sieťového rušenia (jitter) a stratovosti paketov. Udržuje oneskorenie tak nízke, ako je to možné pri zachovaní najvyššej kvality zvuku.

Acoustic Echo Canceler (AEC) - je komponent spracovávajúci signál, ktorý odstráni v reálnom čase akustické echo vyplývajúce zo zvuku, ktorý prichádza do aktívneho mikrofónu.

Noise Reduction (NR) - jedná sa o software založený na spracovaní signálu, ktorý odstraňuje niektoré druhy šumu v pozadí obvykle spojené s bežnou komunikáciou.

1.8.7 VideoEngine

Ide o framework pre prenos videa z kamery do siete a zo siete na obrazovku.

VP8- video kódek z projektu WebM. Dobre sa hodí pre RTC, pretože je určený pre nízku latenciu.

Video Jitter Buffer - Dynamic Jitter Buffer, pomáha zakryť účinky chvenia a stratu paketov na celkovú kvalitu obrazu.

Image enhancements - odstraňuje obrazový šum zo záberu kamery.

2 Asterisk

2.1 Čo je Asterisk

Základnou charakteristikou pobočkovej ústredne Asterisk je že je opensource, čiže voľne šíriteľnou. Je zostrojená v programovacom jazyku C. Dokáže byť plne funkčná na rôznych platformách ako je Linux, Windows, Mac OS, Free BSD a Open BSD. Výhoda Asterisku spočíva aj v otvorenosti zdrojového kódu. Teda kódy si môže ktokoľvek upravovať podľa svojich požiadavkou. Asterisk je skonštruovaný tak aby dokázal spĺňať aj najnovšie funkcie telekomunikačnej sféry. [5, 3]

Asterisk sa uvádza ako privátna pobočková ústredňa (PBX – Private Branch Exchange). PBX medzi sebou prepojuje koncové zariadenia s jedným alebo niekoľkými poskytovateľmi hlasových služieb.

2.2 Verzie Asterisku

V súčasnosti existuje niekoľko dostupných verzií Asterisku medzi ktoré patrí aktuálne posledná Long Term Support verzia 11. Predchodcami tejto verzie boli Long Term Support verzia 1.8 a Standart Support verzia 10.

Jedným z dôvodov rozdielnosti jednotlivých verzií je dĺžka trvania ich podpory. Podpora verzií Long Term Support (LTS) trvá 4 roky plus jeden ďalší rok na údržbu bezpečnostných opráv.

2.2.1 Long Term Support (LTS) verzie

Verzie tohto typu sú vytvárané pre to odvetvie Asterisku, kde sa sústredí hlavne na stabilitu a užívateľské skúsenosti. Medzi posledné verzie LTS patrí:

- Asterisk 1.8 – LTS (posledná verzia 1.8.26.1)
- Certifikovaný Asterisk 1.8 – LTS (posledná verzia 1.8.15 cert5)
- Asterisk 11 – LTS (posledná verzia 11.8.1)
- Certifikovaný Asterisk 11 – LTS (posledná verzia 11.6 cert2)

2.2.2 Standard Support verzie

Verzie typu Štandard sú vytvárané z tých odvetví Asterisku, z ktorých sa preberajú hlavne nové funkcie. Podpora týchto vetví Asterisku trvá kratšiu dobu ako je to u odvetví verzie LTS. Ich podpora sa udáva na jeden rok, pričom sa prihliada aj na časti ohľadom bezpečnosti. To k tomu pridáva ďalší rok.

Verzie typu štandardnej podpory sú:

- Asterisk 12 – Standard (posledná verzia 12.1.1)

2.2.3 Testovacie vydania

Skúšobné vydania sú ukážky z ďalších verzií Asterisk. Každá z nich je v alfa, beta alebo release candidate fáze. Užívateľom, ktorí potrebujú spustiť najnovšiu verziu pre opravu chýb sa odporúča vyskúšať skúšobné správy (test releases) v ich prostredí a poskytnúť spätnú väzbu na sledovanie problémov.

Verzie typu testovacie vydania sú:

- Asterisk 12 – Standard (posledná verzia 12.0.0-rc1)
- Asterisk 11 – LTS (posledná verzia 11.9.0-rc1)
- Asterisk 1.8 – LTS (posledná verzia 1.8.27.0-rc1)

2.3 Asterisk 12

Asterisk 12 je aktuálne posledná verzia, ktorá je stále v čase vývoja. Vývoj tejto verzie bol zameraná na základné architektonické zmeny a bol obohatený o nové funkcie. [4] Tento vývoj zahŕňal v sebe:

- Flexibilnejšie premostenie jadra založené na tzv. Bridging API
- Nové interné správy alebo hlásenia zbernice
- Štandardizovanie a zlepšenie konzistencie AMI
- Pridanie REST rozhrania (ARI)
- Nový ovládač SIP kanálu, chan_pjsip.

Okrem toho, väčšina premostení v Asterisku sa presmerovala do Bridging API využívanou CongBridge. Hlavné zmeny boli vykonané na väčšinu rozhraní v Asterisku. To zahŕňa nielen AMI ale tiež CDR a CEL.

2.3.1 Aplikácie

2.3.1.1 AgentLogin

Spolu s AgentRequest, bola táto aplikácia upravená tak aby nahradila chan_agent. Akt zavolania kanálu aplikáciou AgentLogin umiestnenej v súbore agentov, ktorí môžu byť vyžiadaní aplikáciou AgentRequest.

Schéma agents.conf sa zmenila. Skôr než špecifikujeme agentov na jednom riadku s čiarkou ich oddeľujúcou, každý agent je definovaný v samostatnom kontexte.

Parametre, ktoré boli z agents.conf odstránené sú:

`maxloginretries, autologoffunavail, updatecdr, goodbye, group, recordformat, urlprefix, savecallsin.`

2.3.1.2 ***AgentRequest***

Nová aplikácia, ktorá vyžaduje prihlásenie agenta zo súboru a premostenie žiadajúceho kanálu s kanálom volaným túto aplikáciu.

2.3.1.3 ***BrigdeWait***

Je novou aplikáciou v Asterisku, ktorá má umiestniť volajúci kanál do po zdržujúceho premostenia. Kanál bude potom čakať v tomto po zdržujúcom premostení dovtedy pokiaľ nedôjde k udalosti, ktorá ho odtiaľ odstráni. Kanály zúčastňujúce sa v po zdržujúcom premostení nekomunikujú s inými kanálmi v tom istom po zdržujúcom premostení.

2.3.1.4 ***CongBridge***

Jedná sa o aplikáciu konferenčného premostenia. Všetci účastníci v premostení možno vyhodiť z konferenčnej miestnosti pomocou kanálového parametru all v ConfBridge kick CLI príkazovom riadku. Samotný výpis príkazu CLI confBridge list bol vylepšený. Pri zobrazovaní informácií o konkrétnom premostení, značky budú teraz zobrazované pre konkrétnych užívateľov určujúcich ich vlastnosti.

2.3.1.5 ***Directory***

Ide o pridanie a možnosti, ktorá povoľuje volajúcemu pridať ďalší „alias“ pre užívateľa v adresári. Táto možnosť musí ale byť v spojení s f, l alebo b možnosťou. Tento „alias“ môže byť špecifikovaný vo voicemail.conf.

2.3.1.6 ***DumpChan***

Samotný DumpChan() zobrazuje informácie o kanáli a listovaní v každej premennej kanálu.

Výstup z DumpChan už nezahŕňa polia DirectBridge (priameho premostenia) alebo IndirectBridge (nepriameho premostenia). Namiesto toho, ak sa kanál nachádza v premostení, teda pole BridgeID obsahuje unikátne ID premostenia, v ktorom sa kanál nachádza.

2.3.1.7 ***MeetMe***

Bol pridaný nový parameter n k ochrane aplikácie s odšumovanou funkciou na kanál spájajúci konferencie. Niektoré kanálové ovládače, ktoré sa líšia počtom zvukových vzoriek v hlasovom rámci, budú pociťovať značné problémy kvality ak bude šum pripojený ku kanálu.

2.3.1.8 ***Stasis***

Je novou aplikáciou v Asterisk 12, ktorá preberá kontrolu kanálu volajúceho aplikáciu cez externý systém. Aktuálne, externé systémy narábajú s kanálmi v Stasise cez Asterisk REST rozhranie (ARI).

2.3.2 **Kódeky**

V Astrerisk 12 bola pridaná plná podpora pre kódeky VP8 a Opus.

2.3.3 **CDR (Call Detail Records)**

Významné zmeny boli vykonané v správaní CDR. Jadro CDR bolo efektívnejšie prepísané a postavené na Stasis správe zbernice (Stasis message bus).

CDR budú teraz vytvárané medzi všetkými účastníkmi v premostení. Pre každý pár kanálov v premostení je CDR vytvorená tak aby reprezentovala komunikačnú cestu medzi týmito dvoma koncovými bodmi. To umožňuje koncovému užívateľovi si vybrať kto bude účtovať za čo počas premosťujúcich operácií s viacerými stranami.

Bola pridaná nová vola do cdr.conf, debug, ktorý spôsobuje značne rozsiahlejší výpis z jadra CDR.

2.3.4 **Kanálové ovládače**

Keď je nastavený kanálový ovládač k spúšťaniu variability oneskorenia vyrovnávajúcej pamäte (jitterbuffers), hoci sú v súčasnosti aplikované bezpodmienečne pokiaľ kanál naviaže premostenie.

2.3.4.1 ***chan_bridge***

Agent chan_agent bol vymazaný a nahradený aplikáciami app_agent_pool modulom AgentLogin a AgentRequest. Agenti sú spojený s volajúcimi s použitím novej aplikácie AgentRequest dial plánu.

Možnosť updatecdr bola vymazaná. Pozmeňovanie názvov kanálov na CDR nie je podporované (z dôvodu predstierania iným kanálom, názov kanálu je názov kanálu).Taktiež

bola vymazaná premenná kanálu AGENTUPDATECDR z toho istého dôvodu. Taktiež boli vymazané možnosti konfigurácie endcall a enddtmf. Využíva sa funkcia CHANNEL(DTMF-features) k nastaveniu DTMF funkcií na agenta kanálu pred zavolaním AgentLogin.

2.3.4.2 ***chan_dahdi***

Bol pridaný nový CLI príkaz pri destroy span. Tento príkaz spôsobí zničenie D-kanálu v určitom rozmedzí a jeho B-kanály.

Pre vymazanie alebo vytvorenie väčšieho množstva kanálov boli pridané nové CLI príkazy a to dahdi destroy channels a dahdi create channels.

2.3.4.3 ***chan_iax2***

Ide o pridanie IPv6 podpory. Chan_iax2 nám slúži práve k spojeniu a komunikácii využívajúcej IPv6.

2.3.4.4 ***chan_pjsip***

Jedná sa o nový SIP kanálový ovládač pre Asterisk. Je vybudovaný na základe PJSIP SIP zásobníku. Chan_pjsip obsahuje zbierku zdrojových modulov poskytujúcich funkčnú časť SIP.

2.3.4.5 ***chan_sip***

Bol pridaný ignore_requested_pref. Ide o to, že pokiaľ je zapnutý, tak sa budú využívať preferované kódeky nastavené pre účastníka na druhej strane namiesto požadovaného kódeku.

Taktiež tu bola pridaná možnosť register_retry_403 pre obídenie serverov u ktorých je známe, že chybne odosielajú 403 v odpovedi na platné REGISTER žiadosti a zároveň povoľuje Asterisku pokračovať k nadviazaniu spojenia.

2.3.5 **Funkcie**

2.3.5.1 ***CDR_PROP***

Jedná sa o novú funkciu. Táto funkcia umožňuje nastaviť vlastnosti na aktívnych CDR kanálov. Vlastnosti akceptujú len booleovské hodnoty k ich nastaveniu/zbaveniu na CDR kanálov. Platné vlastnosti zahŕňajú:

- party_a – určí tento kanál preferovaných Strane (Party) A v akomkoľvek CDR medzi dvoma kanálmi. Pokiaľ dva kanály majú vlastnosti nastavené, tak časť, ktorý sa využíva na vytvorenie kanálu je využitý na určenie toho kto bude Stranou (Party) A.
- disable – deaktivuje CDR na kanále.

2.3.5.2 ***JITTERBUFFER***

Už akceptuje argument disabled, ktorý môže byť požitý k vymazaniu predošle nastavených jitterbufferov na kanáli s JITTERBUFFER.

2.3.5.3 ***PJSIP_DIAL_CONTACTS***

Je novou funkciou poskytovanou chan_pjsip. Táto nová funkcia je používaná v spojitosti s aplikáciou Dial k vybudovaniu vytáčaného reťazca, ktorý bude vytáčať každý kontakt v Address of Record (zozname adries) združeného s chan_pjsip koncovým bodom.

2.3.5.4 ***PJSIP_MEDIA_OFFER***

Tak ako predchádzajúca funkcia tak aj táto funkcia je poskytovaná chan_pjsip. Funkcia PJSIP_MEDIA_OFFER nastavuje kódeky, ktoré majú byť ponúkané na smerujúci kanál pred samotným v vytáčaním.

3 Realizácia WebRTC komunikácie na Asterisku

Táto kapitola sa zaoberá praktickým zapojením, konfiguráciou, testovaním voľne šíriteľného projektu WebRTC v spolupráci so softvérovou ústredňou Asterisk a projektom sipML5.

Začneme inštaláciou operačného systému Ubuntu 12.04.4 na virtuálnom stroji. Následne prevedieme inštaláciu telefónnej ústredne Asterisk verzie 11.

3.1 Schéma zapojenia a použité aplikácie

3.1.1 Schéma zapojenia

Praktická časť diplomovej práce bola realizovaná na osobnom počítači a virtuálnom serveri. Vďaka tomuto riešeniu je inštalácia serveru nezávislá na hardware. Celú prácu je taktiež možné zrealizovať na rôznych virtualizačných nástrojoch.

Ako server bol zvolený operačný systém Ubuntu 12.04.4 Server Edition. Ďalej tu bol nainštalovaný Asterisk 11 a to z toho dôvodu, že predchádzajúce verzie Asterisku nepodporujú projekt webrtc. Moja schéma zapojenia je znázornená na obrázku 3.1.

3.1.2 Ubuntu 12.04.4 Server

Ubuntu je operačný systém, ktorý je postavený na základoch Debianu. Debian je známy pre svoje robustné serverové inštalácie. Ubuntu je dostupný k stiahnutiu zdarma a mnou použitá verzia 12.04.4 je zároveň verziou LTS (Long Term Support) s podporou výrobcu až do roku 2017.

Táto distribúcia disponuje obrovským množstvom balíčkov, ktoré nie je potreba kompilovať a je možné ich jednoducho nainštalovať.

3.1.3 Asterisk 11.8.1

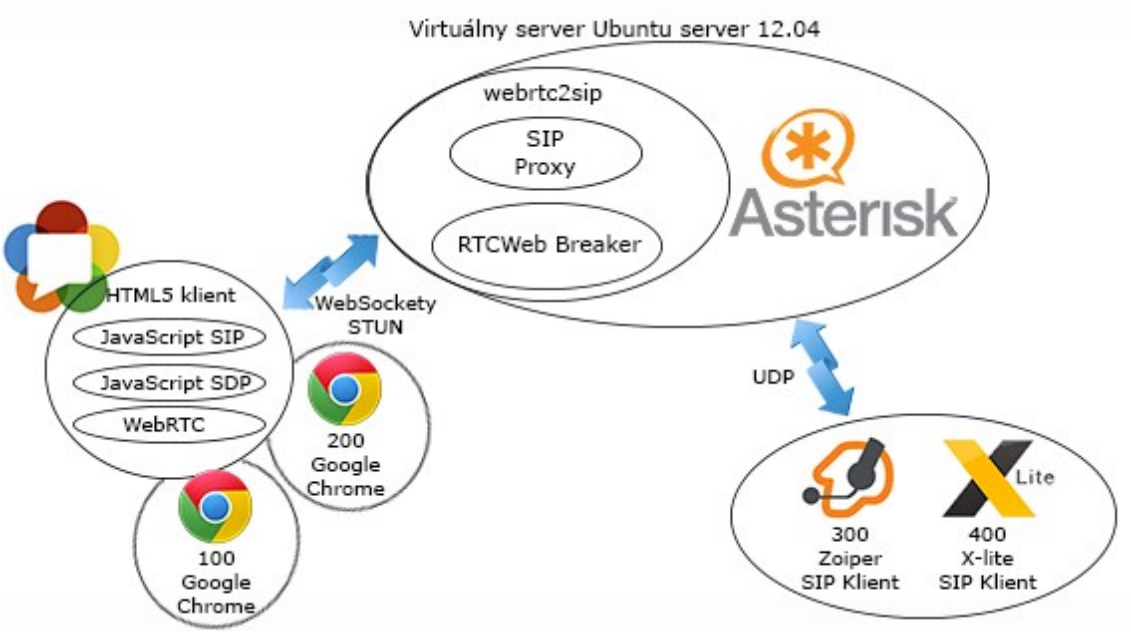
Asterisk je úplná softwarová pobočková ústredňa. Pracuje na operačnom systéme Linux a poskytuje všetky funkcie, ktoré sa od PBX očakávajú. Verzia 11 bola zvolená z toho dôvodu, že od tejto verzie začala podpora technológie WebRTC.

3.1.4 SipML5

SipML5 je prvý voľne šíriteľný HTML5 sip klient, ktorý je celý napísaný v javascripte. Výhodou je, že nie je za potreby žiadne ďalšie rozšírenie, plugin alebo bránu. Samotný klient ho môže využiť k pripojeniu sa k akémukoľvek SIP alebo IMS sieti z mnou preferovaného webového prehliadača k vytvoreniu a prijatiu audio/video hovoru.

Protokolové analyzátory (SIP, SDP ...) sú vysoko optimalizované s použitím Ragel vyhľadávacích tabuliek je vhodný pre vstavané systémy s obmedzenou veľkosťou pamäte a nízkym výpočtovým výkonom. JavaScript SIP/SDP sú napísané v javascripte a k prenosu využívajú Websockety.

Na obrázku 3.1 je znázornená schéma zapojenia sipML5 v zapojení s Asteriskom, ktorú používam v mojej praktickej časti.



Obrázok 3.1: *Schéma zapojenia*

3.1.4.1 webrtc2sip

Je šikovná a výkonná brána využívajúca WebRTC a SIP, k tomu aby premenil webový prehliadač do „telefónu“ so spôsobilosťou na audio, video. Brána umožňuje webovému prehliadaču vytvoriť a prijímať hovory z alebo do SIP siete. Samotná brána pozostáva zo 4 modulov a to sú:

- **SIP Proxy** - úlohou tohto modulu je konvertovanie SIP z WebSocket protokolu do UDP, TCP alebo TLS , ktoré sú podporované každou sieťou. Ide o to ak poskytovateľ siete alebo server poskytuje SIP cez WebSocket (Asterisk,

Kamailio) potom sa môže vynechať/obísť modul a spojiť klienta priamo s koncovým bodom. Obchádzanie SIP Proxy nie je doporučené ak je plánové využitie modulov RTCWeb Breaker-u alebo Media Coder-u a ak je potrebné udržiavať dve rôzne spojenia.

- **RTCWeb Breaker** - zo špecifikácie WebRTC vyplýva povinná podpora pre ICE a DTLS/SRTP. Problémom je, že mnoho koncových bodov SIP toto nepodporuje. A práve kvôli tomu RTCWeb Breaker vyjedná a prekonvertuje dátový tok médií tak aby sa tieto dva „svety" dohodli na vzájomnej spolupráci.
- **Media Coder** - štandard WebRTC definoval dva audio kódeky MTI (Mandatory To Implement – povinné k implementácii) a to opus a g.711. V súčasnosti prebieha intenzívna diskusia ohľadom MTI video kódekov. Vybera sa medzi VP8 a H.264. VP8 je úplne voľne šíriteľný ale nie príliš využívaný zatiaľ čo H.264 AVC nie je voľne šíriteľný ale je používaný vo väčšej miere. Google sa rozhodol pre kódek VP8 v Chrome, Ericsson zas H.264 v Browser-i. Software od Mozilly a Opery bude pravdepodobne využívať VP8 a Microsoft H.264 AVC.
- **Click-to-Call** - jedná sa viac o službu ako modul, kompletný SIP click-to-call projekt založený na 3 iných prvkoch. Cieľom je umožniť osobe prijímať maily, navštevovať webové stránky, sledovať Facebook/Google+ profil k vytvoreniu hovoru na tvoj mobilný telefón pomocou jedného kliku (single click).

3.1.4.2 ***Kompatibilita***

SipML5 v súčasnosti podporuje mnoho webových prehliadačov v rôznych operačných systémoch. Pre viac informácií navštívte oficiálne stránky sipML5 no najviac doporučený je webový prehliadač Google Chrome.

3.2 Asterisk a sipML5

Realizácia je možná dvoma možnými spôsobmi. Jeden sa realizuje cez patch verzie Asterisku. Táto možnosť spôsobuje množstvo problémov a práve aj preto a aj na odporúčanie oficiálnych stránok sipML5 som si vybral možnosť s webrtc2sip. Pred samotnou inštaláciou ústredne Asterisk musíme systém aktualizovať, reštartovať a nainštalovať základné baličky pre správnu inštaláciu, kompiláciu a chod ústredne Asterisk [2, 17].

Na začiatok zdôrazňujem, že jednotlivé príkazy sú vkladané pod administrátorským režimom.

```
sudo -i
```

Pre viac informácií o jednotlivých balíčkoch viz literatúra [4].

```
1. apt-get update && reboot
```

2. `apt-get install build-essential subversion wget libssl-dev libncurses5-dev libnewt-dev libxml2-dev linux-headers-$(uname -r) libsqlite3-dev uuid-dev libtool automake git-core libspeexdsp-dev yasm libvpx-dev libx264-dev screen pkg-config`

Zdôraznil by som, že ešte pred tým než sa pustíme do samotnej konfigurácie a inštalácie Asterisku je dôležité spustiť podporu SRTP, ktorú WebRTC vyžaduje. Toho môžeme docieliť rôznymi spôsobmi. Ja som využil inštaláciu balíčka libsrtp, ktorý popisujem ďalej. Dôležité je aby bola podpora SRTP spustená ešte pred samotnou kompiláciou a inštaláciu Asterisku.

3. `apt-get install libsrtp0-dev`

Ak by sme nemali na Asterisku spustený modul mohlo by dôjsť k zobrazeniu nasledujúcej chyby vo výpise CLI:

```
ERROR[10167]: chan_sip.c:27987 setup_srtp: No SRTP module
loaded, can't setup SRTP session.
```

K správnej činnosti webrtc2sip je potrebné nainštalovať programy ffmpeg a doubango. Doubango je nosnou konštrukciou (framework) na ktorej pracuje webrtc2sip a aby bolo možné pre doubango vykonať potrebný prechod tak potrebujeme knižnice z projektu ffmpeg. [22]

ffmpeg

Pred samotnou kompiláciou a inštaláciou ffmpeg je potrebné si odstrániť existujúce libav balíčky zo systému, ktoré by mohli spôsobiť problémy. Toho docielim nasledovne.

4. `apt-get remove libavutil51`

Po odinštalovaní balíčkov libav začnem s kompiláciou a inštaláciou ffmpeg.

5. `wget -c http://ffmpeg.org/releases/ffmpeg-1.0.2.tar.gz`
6. `tar zxvf ffmpeg-1.0.2.tar.gz`
7. `cd ffmpeg`
8. `./configure --extra-cflags="-fPIC" --extra-ldflags="-lpthread" --enable-pic --enable-memalign-hack --enable-shared --disable-static --disable-network --disable-protocols --disable-pthreads --disable-devices --disable-filters --disable-bsfs --disable-muxers --disable-demuxers --disable-parsers --disable-hwaccels --disable-ffmpeg --`

```
disable-ffplay --disable-ffserver --disable-encoders --
disable-decoders --disable-zlib --enable-gpl --disable-
debug --enable-encoder=h263 --enable-encoder=h263p --
enable-decoder=h263 --enable-encoder=mpeg4 --enable-
decoder=mpeg4 --enable-libx264 --enable-encoder=libx264 --
enable-decoder=h264
```

9. `make && make install`
10. `ldconfig`

Doubango

Nasledujúcimi príkazmi spustíme inštaláciu doubango frameworku zo zdrojových balíčkov a využijem k tomu subversion.

11. `svn co http://doubango.googlecode.com/svn/branches/2.0/doubango doubango`
12. `cd doubango`
13. `sed -i '1,/==/s/==/=/' autogen.sh`
14. `./autogen.sh`
15. `./configure --with-ssl --with-srtp --with-vpx --with-speex --with-speexdsp --enable-speexresampler --enable-speexjb --enable-speexdenoiser --with-ffmpeg --with-h264 --prefix=/usr/local`
16. `make && make install`
17. `ldconfig`

webrtc2sip

Po úspešnom nainštalovaní doubango, prejdeme k stiahnutiu zdrojových súborov webrtc2sip taktiež pomocou subversion a následne ich skompilujeme.

18. `svn co http://webrtc2sip.googlecode.com/svn/trunk/ webrtc2sip`
19. `cd webrtc2sip`
20. `sed -i '1,/==/s/==/=/' autogen.sh`
21. `./autogen.sh`
22. `./configure --with-doubango=/usr/local --prefix=/usr/local`

```
23.    make && make install
24.    mkdir -p /usr/local/etc/webrtc2sip
25.    cp config.xml /usr/local/etc/webrtc2sip/
```

Pokiaľ sa nám konfigurácia webrtc2sip podarila jej konfiguračný súbor nájdeme v /usr/local/etc/webrtc2sip/config.xml. Spustenie webrtc2sip balíčkov urobíme nasledovne s výpisom.

```
26.    screen -dmS webrtc2sip webrtc2sip
       --config=/usr/local/etc/webrtc2sip/config.xml && screen
       -r webrtc2sip
```

3.2.1 Inštalácia softwarovej ústredne Asterisk

Asterisk 11 povoľuje registrovanie sa a vytváranie hovorov z webového prehliadača Chrome. Problém je, že Asterisk 11 neobsahuje v sebe podporu video kódeku VP8. To som vyriešil samotnou implementáciu kódeku z externého zdroja. Patch poskytuje ale len experimentálnu podporu video kódeku VP8.

V mojom prípade som si použil verziu Asterisku 11.1.2

```
27.    cd /usr/src/
28.    wget
       http://downloads.asterisk.org/pub/telephony/asterisk/
       old-releases/asterisk-11.1.2.tar.gz
       tar zxvf asterisk-11.1.2.tar.gz
       cd asterisk-11.1.2
```

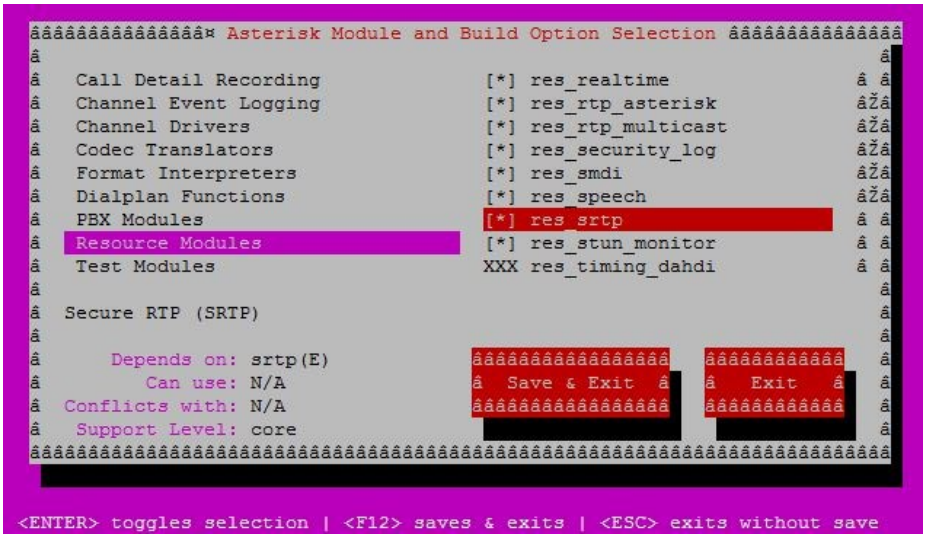
Podpora video kódeku VP8

Vieme, že WebRTC má podporu video kódeku VP8 no musíme si pripomenúť, že podpora video kódeku sa všade uvádza ešte ako experimentálna no aj tak sa túto možnosť pokúsime využiť za pomoci patchu, ktorý si stiahneme do rozbaleného súboru /asterisk-11.1.2.

```
29.    wget https://raw.github.com/AutoStatic/asterisk-
       vp8/master/asterisk_11_vp8_passthrough_support.patch
30.    patch -p1 -i
       asterisk_11_vp8_passthrough_support.patch
31.    ./configure --with-crypto --with-ssl --with-srtp
```

V tomto bode je dobré si prezrieť nastavenia, ktoré budú následne skompilované. Všímame si hlavne zaznačenie res_srtp modulu ako je vidieť na obrázku 3.2.

```
32.     make menuselect
```



Obrázok 3.2: *Moduly softwarovej ústredne Asterisk*

Ďalej pokračujeme v inštalácii softwarovej ústredne Asterisk.

```
33.     make && make install && make samples
```

Po nainštalovaní Asterisku začneme s konfiguráciou konfiguračných súborov.

```
34.     cd /etc/asterisk/
```

Pre udelenie prístupu účastníkom sa použije prenos cez WebSocket, ktorý musí byť rozšírený o možnosti prenosu. V mojom prípade udp, ws. V konfiguračnom súbore sip.conf nastavíme.

```
35.     udpbindaddr = 0.0.0.0
36.     realm = 10.0.1.238
37.     transport = udp,ws
38.     allow = !all,alaw,ulaw,vp8
39.     nat = no
40.     nat = force_rport
41.     nat = auto_force_rport,auto_comedia
```

```
42.     videosupport = yes
```

Vytvorím si šablónu „webrtc-template“ z ktorej budú čerpať ostatní klienti.

```
43.     [webrtc-template](!)
        type = friend
        context = webrtc
        host = dynamic
        allow = h264,h263p,vp8
```

Konfigurácia užívateľov v konfiguračnom súbore sip.conf. Užívateľ WebRTCClient1 a WebRTCClient2 sú webový klienti čiže sipML5 a SIPClient1 SIPClient2 sú SIP klienti.

```
44.     [WebRTCClient1](webrtc-template)
        callerid = "WebRTCClient1" <100>
        secret= 123456
        directmedia = outgoing

        [WebRTCClient2](webrtc-template)
        callerid = "WebRTCClient2 <200>
        secret=456789
        directmedia = outgoing

        [SIPClient1](webrtc-template)
        callerid = "SIPClient1" <300>
        secret=123456
        directmedia = no
        mailbox=default

        [SIPClient2](webrtc-template)
        callerid = "SIPClient2" <400>
        secret=456789
        directmedia = no
```

```
mailbox=default
```

Konfigurácia dial plánu pre každého účastníka ústredne v konfiguračnom súbore extensions.conf, ktoré zapíšeme do sekcie „public“ a „webrtc“.

45.
```
[public]
exten => s,1,NoOp()
 same => n,Hangup()

[webrtc]
exten => 100,1,Dial(SIP/WebRTCClient1)
 same => n,Hangup()

exten => 200,1,Dial(SIP/WebRTCClient2)
 same => n,Hangup()

exten => 300,1,Dial(SIP/SIPClient1)
 same => n,Hangup()

exten => 400,1,Dial(SIP/SIPClient2)
 same => n,Hangup()
```

Po nastavení konfiguračných súborov spustíme softwarovú ústredňu Asterisk.

46.
```
Asterisk -rvvv
```

3.2.2 sipML5

Webový prehliadač Google Chrome

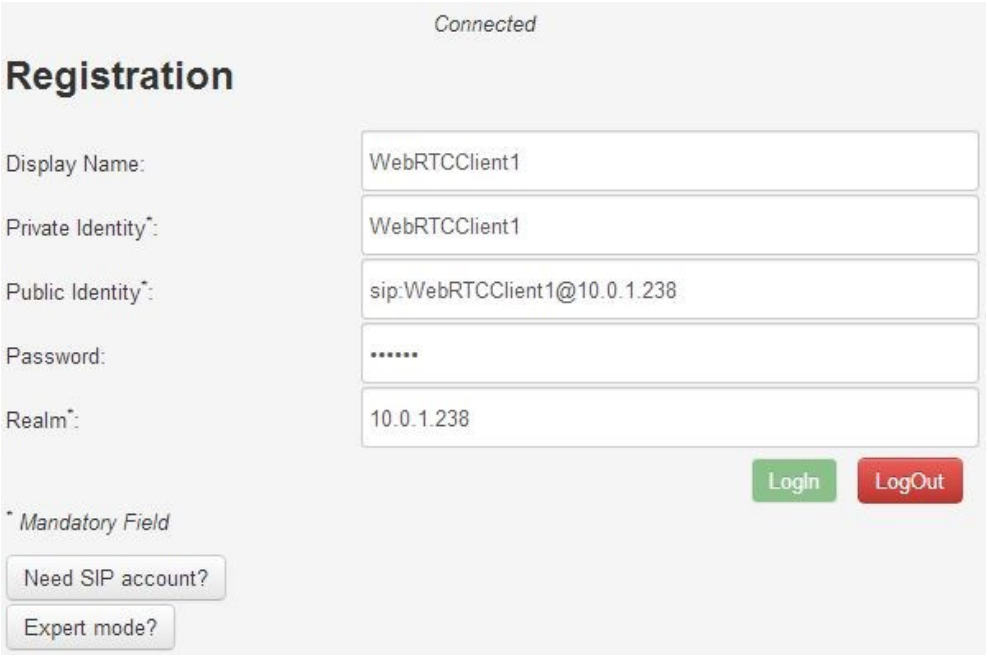
V mojej práci som si vybral webový prehliadač Google Chrome verzie 34.0.1847.131 m. Je to prehliadač, ktorý v sebe spája minimalistický dizajn a dômyselnú technológiu pre rýchlejší, bezpečnejší a jednoduchší web[23]. Tento prehliadač podporuje najviac technológia WebRTC.

Pomocou webového prehliadača, v mojom prípade Google Chrome som prešiel na webovú adresu sipML5.org.

```
47.      http://sipml5.org/call.htm
```

Rád by som spomenul, že je možné si sipML5 klienta stiahnuť aj priamo do softvérovej ústredne Asterisk. Túto možnosť som vyskúšal ale táto možnosť spôsobuje množstvo problémov..

Zobrazilo sa nám registračné menu webového klienta sipML5 a vyplníme nasledovné kolónky podľa obrázka 3.3 . Po vyplnení a ešte pred pripojením sa prepneme do módu expert a nastavíme, vyplníme podľa obrázka 3.4, následne uložíme. Uložením v nastaveniach „expert“ sa pripojíme pomocou tlačidla „LogIn“ po čom sa nám klient „WebRTCClient1“ pripojí. Pripojenie dokazuje zobrazenie stavu „Connected“ v registračnom menu a výpis o pripojení v softwarovej ústredni Asterisk.



Obrázok 3.3: *Nastavenie registračného menu sipML5*

V nastaveniach expert je dôležité aby sme mali zaškrtnutú možnosť RTCWeb Breaker. Ako som spomínal vyššie RTCWeb Breaker vyjednáva a prekonvertuje dátový tok médií tak aby sa dva „svety“ dohodli na vzájomnej spolupráci.

Obrázok 3.4: *Expert nastavenie v simpl5*

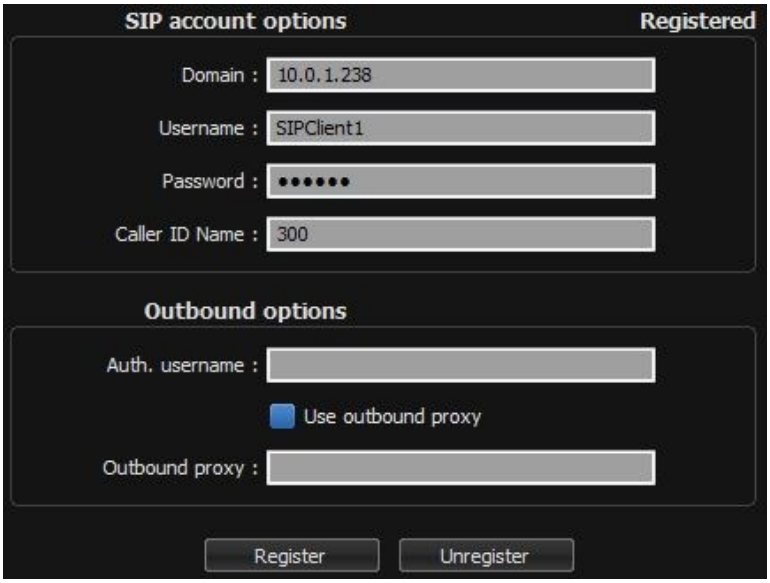
Výpis zo softwarovej ústredne Asterisk o zaregistrovaní sipML5 klienta „WebRTCClient1“.

```
-- Registered SIP 'WebRTCClient1' at 10.0.1.238:10060
```

Konfigurácia Zoiper SIP klienta

Zoiper je VoIP softwarový telefón, ktorý umožňuje komunikáciu pomocou hlasu a videa. Na rozdiel od iného softwaru ako napríklad Skype alebo Viber je Zoiper otvorený a môže sa používať s akýmkoľvek VoIP poskytovateľom alebo PBX ústredňou. Podporuje širokú škálu kódekov. U audia sú to napríklad Speex, a-law, u-law a u videa sú to napríklad H263 Plus a VP8. V mojej diplomovej práci využívam Zoiper klienta verzie 3.2.21357.

Konfigurácia Zoiper klienta mnou je na obrázku 3.5.



Obrázok 3.5: *Konfigurácia Zoiper VoIP klienta*

Výpis zo softwarovej ústredne Asterisk o zaregistrovaní SIP klienta „SIPClient1“.

```
-- Registered SIP 'SIPClient1' at 10.2.3.18:53890
```

X-lite SIP klient

Jeho konfigurácia prebiehala podobne ako konfigurácia Zoiper SIP klienta. X-lite je ako aj Zoiper komerčný softvérový telefón podporujúci SIP protokol. V práci som použil verziu 4.0 [24].

4 WebRTC kvalitatívne parametre a požiadavky na sieť

Dalo by sa povedať, že požiadavky na sieť sa líšia v závislosti od toho aké audio a video kódeky budeme využívať. Ako je spomenuté vyššie tak WebRTC využíva tri audio kódeky a jeden video kódek. [19, 16]

Je treba ale upozorniť na to, že to neznamená využívanie len týchto kódekov ak budeme používať technológiu WebRTC a tiež tieto kódeky nemusia pracovať zatiaľ bezchybne preto sa využívajú aj iné kódeky.

4.1 Audio kódeky

- **iSAC –** je širokopásmový a super širokopásmový audio kódek. iSAC využíva 16 kHz alebo 32 kHz vzorkovaciu frekvenciu s adaptívnym a variabilným dátovým tokom o veľkosti 12 alebo 52 kbps.
- **iLBC –** je úzkopásmový audio kódek. Využíva 8 kHz vzorkovaciu frekvenciu s dátovým tokom 15,2 kbps pre 20 ms rámce a 13,33 kbps pre 30 ms rámce. Je definový v IETF RFC 3951 a 3952.
- **Opus –** podporuje konštantné a variabilné kódovanie dátového toku od 6 kbit/s do 510 kbit/s, pre veľkosti rámcov od 2,5 ms do 60 ms a rôzne vzorkovania od 8 kHz (s 4 kHz šírkou pásma) do 48 kHz (s 20 kHz šírkou pásma). Je definovaný v IETF RFC 6176.

4.2 Video kódek

- **VP8 –** pri hovore one-to-one, potrebujú byť kóder aj dekóder schopný spracovať 720 pixelov s rýchlosťou 30 fps (rámcov za sekundu) pri šírke pásma 2 Mbps.

Avšak web Asterisk.org uvádza že v spojení s technológiou WebRTC podporuje aj kódeky g.711, g722. [1,8]

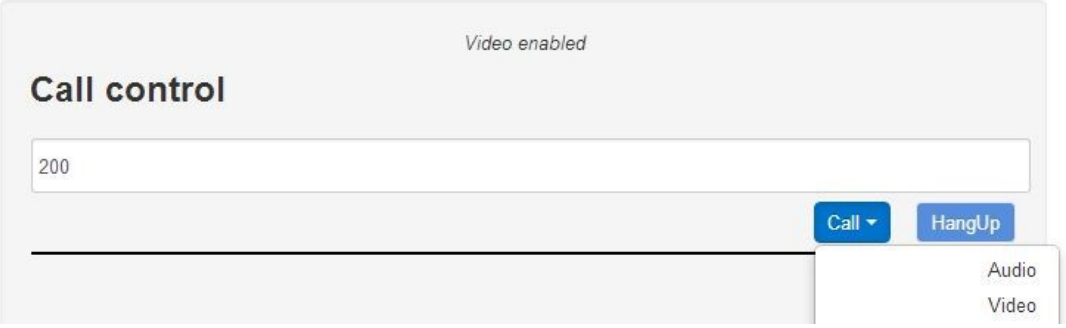
- **g.711** – je jedným z najviac používaných kódekov. Je širokopásmový audio kódek určený pre VoIP komunikáciu. Využíva 8 kHz vzorkovaciu frekvenciu pre šírku pásma 64 kbps.
- **g.722** – je taktiež širokopásmový audio kódek určený pre VoIP komunikáciu. Využíva 16 kHz vzorkovaciu frekvenciu pre šírku pásma 48, 56 a 64 kbps.

Meraním som zistil, že pre minimálnu funkčnosť video hovoru by sme mali použiť spojenie s minimálnou šírkou pásma 300 kbps. Pri nižších hodnotách sa spojenie začalo rozpadať.

5 Analýza WebRTC komunikácie

5.1 Vytvorenie spojenia

Po konfigurácii a prihlásení oboch klientov do softwarovej ústredne Asterisk, vytvorím spojenie medzi sipML5 a Zoiper VoIP klientom. V sipML5 klientovi zadám do ovládania hovoru „Call control“ číslo Zoiper VoIP klienta „300“ a vytvorím „Audio“ hovor ako je znázornené na obrázku 5.1.



Obrázok 5.1: *Call control sipML5*

Po chvíľke nám začne zvoniť „SIPClient1“ Zoiper VoIP klient, hovor preberieme. V softwarovej ústredni sa nám zobrazí nasledujúci výpis, ktorý popisuje celé spojenie.

```
       == Using SIP VIDEO CoS mark 6
       == Using SIP RTP CoS mark 5
         -- Executing [300@webrtc:1] Dial("SIP/WebRTCClient1-
00000002", "SIP/SIPClient1") in new stack
       == Using SIP VIDEO CoS mark 6
       == Using SIP RTP CoS mark 5
         -- Called SIP/SIPClient1
         -- SIP/SIPClient1-00000003 is ringing
         -- SIP/SIPClient1-00000003 answered SIP/WebRTCClient1-
00000002
         -- Locally bridging SIP/WebRTCClient1-00000002 and
SIP/SIPClient1-00000003
  == Spawn extension (webrtc, 300, 1) exited non-zero on
'SIP/WebRTCClient1-00000002'
```

5.1.1 Analýza spojenia medzi sipML5 klientom a zoiper SIP klientom

Celý dialóg si za pomoci programu Wireshark odchytíme. Pre názornejšie znázornenie dialógu využijeme analýzu grafov poskytnutého programom Wireshark.

Zo strany SIPClienta1 prebiehala komunikácia klasicky ako je nám známe z literatúry [20]. Zo strany sipML5 klienta prebieha spojenie pomocou websocketov a STUN.

Websocket [13]

Je protokol poskytujúci obojsmerné spojenie (full-duplex) komunikačných kanálov cez jedno TCP spojenie. Websocket je navrhnutý tak aby bol implementovaný vo webových prehliadačoch a webových serveroch, samozrejme môže byť použitý na klientskych a serverových aplikáciách. Spojenie s ústredňou teda prebieha cez protokol TCP z pohľadu webového prehliadača.

Spojenie medzi sipML5 klientom a softwarovou ústredňou Asterisk pomocou websocketov prebehne nasledovne.

Žiadosť od webového klienta na server. V žiadosti žiada klient prepnutie (switch) protokolu HTTP na Websocket. Toto spojenie sa tiež nazýva ako Websocket handshake.

```
GET / HTTP/1.1
Upgrade: websocket
Connection: Upgrade
Host: 10.0.1.238:10060
Origin: http://sipml5.org
Sec-WebSocket-Protocol: sip
Pragma: no-cache
Cache-Control: no-cache
Sec-WebSocket-Key: 3B4b3HpAyO5fUbwK8YrS8Q==
Sec-WebSocket-Version: 13
Sec-WebSocket-Extensions: permessage-deflate;
```

Odpoveď softvérovej ústredne. Softvérová ústredňa porozumela Websocket protokolu cez hlavičku Upgrade.

```
HTTP/1.1 101 Switching Protocols
Content-Length: 0
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: VlbftaP/Yr5rjgGwtHxhvZpKcUk=
Sec-WebSocket-Protocol: sip
Sec-WebSocket-Version: 13
```

Spojenie so softvérovou ústredňou Asterisk sa podobá protokolu HTTP, čiže server zvládne pripojenie websocketu na rovnakom porte ako HTTP. V tomto bode sa ale HTTP spojenie rozpadne a nahradí sa Websocket spojením, ktorý pracuje na TCP/IP spojení. Pre viac informácií viz. literatúra [15].

Kód napísaný v javascripte vytvára zabezpečené websocket spojenie so SIP Proxy/Registrar, ktorý je na softvérovej ústredni Asterisk. Počas vytvorenia websocket spojenia, webový klient „100“ skonštruuje, odošle SIP REGISTER so žiadosťou na Outbound. Vzhľadom k tomu, že javascript zásobník v prehliadači nemá žiadny spôsob ako určiť adresu z ktorej je websocket spojenie vytvorené, táto implementácia využíva náhodný „.invalid“ názov domény pre Via odosielateľa a ako URI v Kontakt záhlaví [15].

Správa REGISTER:

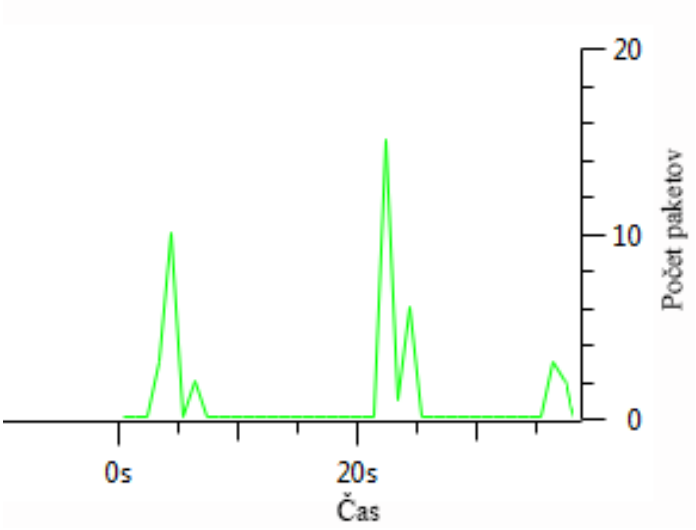
```
REGISTER sip:10.0.1.238 SIP/2.0
Via: SIP/2.0/WS df7jal23ls0d.invalid;
branch=z9hG4bK3em06IJGK3aSHIs2Cgp1UruuJZu7MbKw;rport
From: "WebRTCClient1"<sip:WebRTCClient1@10.0.1.238>;
tag=2eTeK9W0aNJbh9AwigAH
To: "WebRTCClient1"<sip:WebRTCClient1@10.0.1.238>
Contact:
"WebRTCClient1"<sip:WebRTCClient1@df7jal23ls0d.invalid;rtcweb-
breaker=yes;transport=ws>;expires=200;click2call=no;+g.oma.sip-
im;+audio;language="en,fr"
Call-ID: 82d648a6-28ee-1379-4624-5e25c0633dec
CSeq: 58278 REGISTER
Content-Length: 0
Max-Forwards: 70
```

```
Authorization: Digest
username="WebRTCClient1",realm="10.0.1.238",nonce="38a1b2f6",uri
="sip:10.0.1.238",response="f53ec1512a20e86e825e44ad44b0d7e9",al
gorithm=MD5

User-Agent: IM-client/OMA1.0 sipML5-v1.2014.04.18

Organization: Doubango Telecom

Supported: path
```

Odpoveď zo strany serveru je nasledujúca:

```
SIP/2.0 200 OK

Via: SIP/2.0/UDP
10.0.1.238:10060;rport=10060;received=10.0.1.238;
branch=z9hG4bK3em06IJGK3aSHIs2Cgp1UruuJZu7MbKw

From:
"WebRTCClient1"<sip:WebRTCClient1@10.0.1.238>;tag=2eTeK9W0aNJbh9
AwigAH

To: "WebRTCClient1"<sip:WebRTCClient1@10.0.1.238>;tag=as5b3805e2

Contact: <sip:WebRTCClient1@10.0.1.238:10060;rtcweb-
breaker=yes;transport=udp;ws-src-ip=10.2.3.18;ws-src-
port=2732;ws-src-proto=ws>;expires=200

Call-ID: 82d648a6-28ee-1379-4624-5e25c0633dec

CSeq: 58278 REGISTER

Expires: 200

Content-Length: 0


Via: SIP/2.0/TCP
10.2.3.18:2732;rport;branch=z9hG4bK3em06IJGK3aSHIs2Cgp1UruuJZu7M
bKw;ws-hacked=WS

Server: Asterisk PBX 11.1.2

Allow:
INVITE,ACK,CANCEL,OPTIONS,BYE,REFER,SUBSCRIBE,NOTIFY,INFO,PUBLIS
H
```

Protokol TCP po ktorom sa nadväzuje spojenie so softwarovou ústredňou a ktorý využívajú websockety je graficky znázornení na obrázku 5.2. Z jeho zobrazenia je vidieť, že TCP protokol zaťažuje sieť len minimálne a len v bodoch prihlásenia, vytvorenia a ukončenia spojenia sipML5 klienta so softwarovou ústredňou Asterisk.



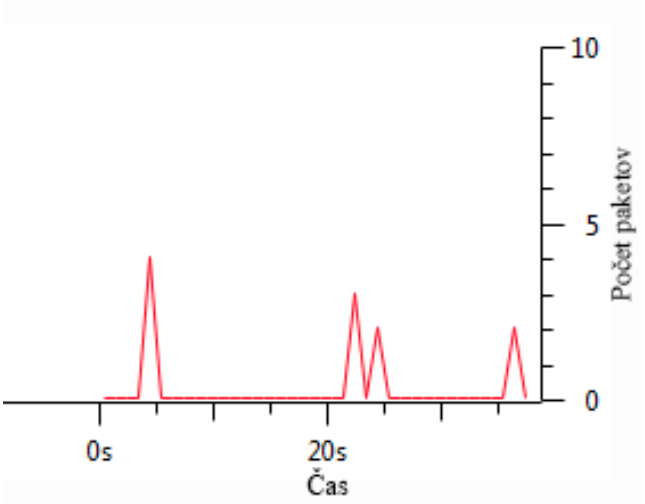
Obrázok 5.2: *Grafické zobrazenie zaťaženia TCP paketmi na sieť pri spojení sipML5 klienta a SIP klienta*

Samotné dĺžky paketov sa pohybujú v rozmedziach ktoré sú zaznamenané v tabuľke 5.1.

Tabulka 5.1: *Dĺžky TCP paketov pri spojení sipML5 klienta a SIP klienta*

Dĺžky paketov	**Množstvo**	**Percentuálny podiel**
40-79	16	38,10%
80-159	3	7,14%
160-319	9	21,43%
320-639	6	14,39%
640-1279	5	11,90%
1280-2559	3	7,14%

Na obrázku 5.3 zo spojenia môžeme vidieť zaťaženie websocketmi na sieť.



Obrázok 5.3: *Grafické zobrazenie zaťaženia websocketmi na sieť pri spojení sipML5 klienta a SIP klienta*

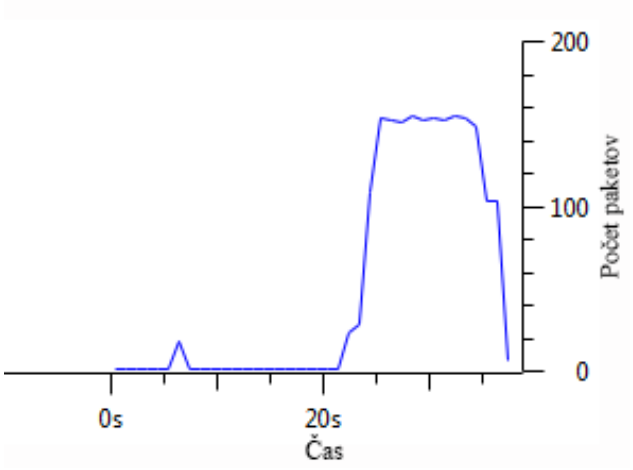
Z obrázku 5.3 je vidieť, že websockety zaťažujú sieť len nepatrne počas celého trvania spojenia a dialógu. Samotné dĺžky paketov sa pohybujú v rozmedziach ktoré sú zaznamenané v tabuľke 5.2.

Tabulka 5.2: *Dĺžky websocket paketov pri spojení sipML5 klienta a SIP klienta*

Dĺžky paketov	**Množstvo**	**Percentuálny podiel**
320-639	5	45,45%
640-1279	5	45,45%
1280-2559	1	9,09%

Po spojení si môžu vymieňať informácie oba konce v reálnom čase. Následný dialóg zo strany sipML5 klienta so softwarovou ústredňou pokračuje za pomoci STUN. STUN je komunikácia za pomoci UDP paketov na náhodne vybraných portoch, ktoré sú aktuálne voľné. V našom prípade sa jedná o čísla portov 60918 čo je port na strane sipML5 klienta a 40843 je port na strane ústredne. Tieto porty boli vyjednané STUN.

Zaťaženie UDP paketmi na sieť je už omnoho väčšie ako je vidieť na obrázku 5.4 pretože sa prenášajú dáta, samotný hovor.



Obrázok 5.4: *Grafické zobrazenie zaťaženia UDP paketmi na sieť pri spojení sipML5 klienta a SIP klienta*

Je vidieť, že zaťaženie zo strany UDP paketov je už znateľnejšie, pričom dĺžky paketov sa pohybujú v rozmedziach ktoré sú zobrazené v tabuľke 5.3.

Tabulka 5.3: *Dĺžky UDP paketov pri spojení sipML5 klienta a SIP klienta*

Dĺžky paketov	Množstvo	Percentuálny podiel
40-79	1	0,05%
80-159	33	1,73%
160-319	1835	96,22%
320-639	19	1,00%
640-1279	18	0,94%
1280-2559	1	0,05%

Analýza RTP toku z pohľadu sipML5 klienta

Zo strany sipML5 klienta neuvidíme žiaden RTP tok, kvôli tomu že technológia WebRTC dbá veľmi na bezpečnosť tak v paketovom analyzátore neuvidíme žiaden tok. Na toto je veľmi dôležité poukázať pretože toto je tiež jedným z rozdielov oproti stávajúcej technológii.

Analýza RTP toku z pohľadu SIPClient1 klienta

Na strane SIPClient1 klienta si dokážeme odchytiť tok RTP paketov.

Meranie RTP toku si predvedieme v analyzátore RTP toku vo Wiresharku na jednom zariadení, pričom aj oba klienti sú na jednom zariadení a to z toho dôvodu aby sme mohli presne stanoviť časy prenosu paketov. V takomto prípade nemusíme využívať protokol, ktorý by synchronizoval čas na rôznych zariadeniach zapojených v sieti.

RTP tok je zachytený v oboch smeroch, to znamená v smere od SIPClient1 klienta k softwarovej ústredni a v smere k SIPCient1 klientovi od ústredne. RTP analýza nám poskytuje detailnejší pohľad na tieto merané veličiny:

- **Max delta –** delta je časový rozdiel medzi predchádzajúcim a aktuálnym paketom v toku. A max delta je ta najdlhšia hodnota delta s presným určením paketu.
- **Max/Mean jitter –** jitter sa uvádza ako variabilita oneskorenia a je dôležitým parametrom hlavne pri prenose hlasu a videa v reálnom čase. Je žiaduce aby bol minimálny, t.j. aby prenosové oneskorenia rôznych paketov boli rovnaké. Čiže max jitter je maximálna variabilita oneskorenia. Pre viac informácií o jej výpočte viz. literatúra [12]. Hodnota mean jitter je aritmetický priemer všetkých hodnôt oneskorenia (jitter).
- **Max Skew (vychýlenie, odchýlenie) –** Skew je meraná veličina toho o koľko je včasný alebo o koľko sa oneskorí daný paket oproti celkovej konverzácii. Skew je kladná pokiaľ timestamp (časová značka) menovitého času (nominal time) je rýchlejšia (väčšia) než čas príchodu paketu (arrival time) [18].

```
statinfo->skew    = nominaltime - arrivaltime;
absskew = fabs(statinfo->skew);
if(absskew > fabs(statinfo->max_skew)){
        statinfo->max_skew = statinfo->skew;
```

  V jednoduchosti napríklad ak bude priemerná rýchlosť paketu 50 paketov za sekundu a povedzme, že tisíci paket spojenia dorazí 20,03 sekundy po prvom pakete, tak potom skew pre tento paket bude -30 ms. A max skew je teda logický najdlhšia hodnota zo skew hodnôt.
- **Total/Lost RTP pakety –** Total uvádza celkové množstvo prenesených RTP paketov a Lost uvádza celkové množstvo stratených paketov pri prenose.
- **Duration –** je dĺžka trvania spojenia v sekundách

V smere od SIPClient klienta (Forward direction)

Max delta = 21,05 ms (paket číslo 1720)

Max Jitter = 0,56 ms Mean Jitter = 0,23 ms

Max skew = - 1,25 ms

Total RTP pakety = 514 Lost RTP pakety = 0 (0,00%)

Duration = 10,26 sekúnd

V smere od SIPClient1 klienta nám vyšli veľmi priaznivé hodnoty kedy maximálny rozdiel medzi dvoma paketmi bol len 21,15 ms s variabilným oneskorením len 0,56 ms. Počas 10,26 sekúnd bolo prenesených 514 RTP paketov, pričom žiaden z nich nebol stratený.

V smere k SIPClient klientovi (Reversed direction)

Max delta = 87,74 ms (paket číslo 1072)

Max Jitter = 19,71 ms Mean Jitter = 14,19 ms

Max skew = -67,99 ms

Total RTP pakety = 604 Lost RTP pakety = 0 (0,00%)

Duration = 12,10 sekúnd

V opačnom smere teda k SIPClient1 klientovi sú merané veličiny dosť rozdielne oproti predchádzajúcemu. No aj napriek tomu boli všetky RTP pakety (604) prenesené a stratovosť bola nulová.

Samotný dialóg prebiehal nasledovne.

INVITE

Samotnú správu INVITE je možné vidieť v prílohe B.1.

- Ako o prvú správu sa jedná o správu INVITE. INVITE sa v tomto prípade využíva k zostaveniu spojenia. URI na prvom riadku *300@10.0.1.238* sa nazýva Request URI a obsahuje URI ďalšieho skoku správy. V tomto prípade bude hostiteľom 10.0.1.238 a hľadá sa užívateľ 300 čo je SIPClient1 klient.
- V poli Via môžeme vidieť, že odpoveď bude doručená na „df7jal23ls0d.invalid“ a k spojeniu sú využívané websockety. Dôvod kvôli čomu je v hlavičke „df7jal23ls0d.invalid“ je vysvetlený v literatúre [12].

- Polia hlavičiek *From* a *To* identifikujú iniciátora a príjemcu spojenia. Parameter *tag* označuje identifikátor dialógu.
- *Call-ID* je identifikátorom dialógu, ktorého cieľom je identifikovať správy náležiace jednému volaniu.
- Pole *Contact* obsahuje IP adresu a port, na ktorom odosielateľ očakáva ďalšie žiadosti odosielané volaným. To znamená, že obe strany si vymenia svoje kontakty v žiadosti a odpovedi. Teda pokiaľ bude chcieť jedna zo strán napríklad ukončiť spojenie pomocou správy BYE, tak v tom prípade nemusí posielať požiadavku na SIP Proxy, ale pošle ju priamo.
- Pomocou poľa *CSeq* sú v dialógu očíslované jednotlivé žiadosti. Svoje využitie nachádza pri odosielaní správ nespoľahlivým prenosom, kde môže dochádzať k opakovaniam a tým pádom príjemca môže detekovať opakovanie a správne tak selektovať žiadosti.

V rovnakom čase prichádza na SIPClient1 klienta takisto správa INVITE. Ktorá má podobný charakter ako správa INVITE pochádzajúca od sipML5 klienta. WebRTCClient1 klientom sa tam označuje sipML5 klient.

TRYING

Správa TRYING je odosielaná z transakčnej vrstvy. Správu TRYING je možné vidieť v prílohe B.2.

- Po tom ako INVITE dorazil na SIP Proxy 10.0.1.238, tak hneď sa odosiela 100 TRYING. V odpovedi 100 TRYING budú rovnaké hodnoty parametrov *To, From, Call-ID, CSeq* ako v prijatom INVITE.

RINGING

- Nasleduje odpoveď 180 RINGING, odosielanú z 300 SIPClient1 teda SIP klientovho UA. UA vyzváňa zároveň odosiela správu 180 RINGING. V hlavičke sa pripíše *received* do *Via* a pridá sa *tag* do poľa *To*. Pred zostavením dialógu sú už definové *from Tag, to Tag, Call-ID*. Odpoveď 180 RINGING sa pre posielaním cez SIP Proxy 10.0.1.238 dostane až k 100 číže WebRTCClient1 klientovi.

Priebeh na správy 180 Ringing na strane WebRTCClient1 je vidieť v prílohe B.3.

200 OK

- Keď 300 SIPClient1 príjme volanie, tak SIPClient1-ov UA odošle odpoveď 200 OK, ktorá je veľmi podobná ako odpoveď 180 RINGING. Odpoveď uzaviera

transakciu medzi 300 SIPClient1-om, SIP Proxy 10.0.1.238 a medzi SIP Proxy 10.0.1.238 a 100 WebRTCClient1 klientom.

Odpoveď 200 OK na strane sipML5 klienta je možné vidieť v prílohe B.4.

ACK

Žiadosť ACK je možné vidieť v prílohe B.5.

- Žiadosť ACK berie 100 WebRTCClient1 klient ako potvrdenie odpovedi 200 OK.

BYE

- Hovor je ukončený SIPClient1-om pomocou žiadosti BYE. Žiadosť BYE sa považuje za novú transakciu, čiže sa zvýši *CSeq* ale *From tag, To tag* a *Call-ID* zostanú zachované.
- Dialóg sa ukonči finálnou odpoveďou 200 OK na BYE.

Žiadosť BYE je možné vidieť v prílohe B.6.

200 OK

- Dialóg sa ukonči finálnou odpoveďou 200 OK na BYE.

Správu 200 OK je možné vidieť v prílohe B.7.

5.1.2 Analýza audio hovoru medzi dvoma sipML5 klientmi

Toto spojenie som inicioval medzi dvoma sipML5 klientmi presnejšie medzi WebRTCClient1 klientom a WebRTCClient2 klientom. Ich konfigurácia bola spomenutá vyššie. V spojení bol použitý kódek ulaw g.711. Bohužiaľ ako som zistil nové kódeky nepracovali správne a spôsobovali problémy s kompatibilitou s webovým prehliadačom.

Spojenie prebiehalo rovnako ako v predchádzajúcom prípade na strane WebRTCClient1 klienta. Najprv som pristúpil k registrácii najskôr na strane sipML5 klienta „WebRTCClienta1“:

```
-- Registered SIP 'WebRTCClient1' at 10.0.1.238:10060
```

Potom aj na strane sipML5 klienta „WebRTCClient2“:

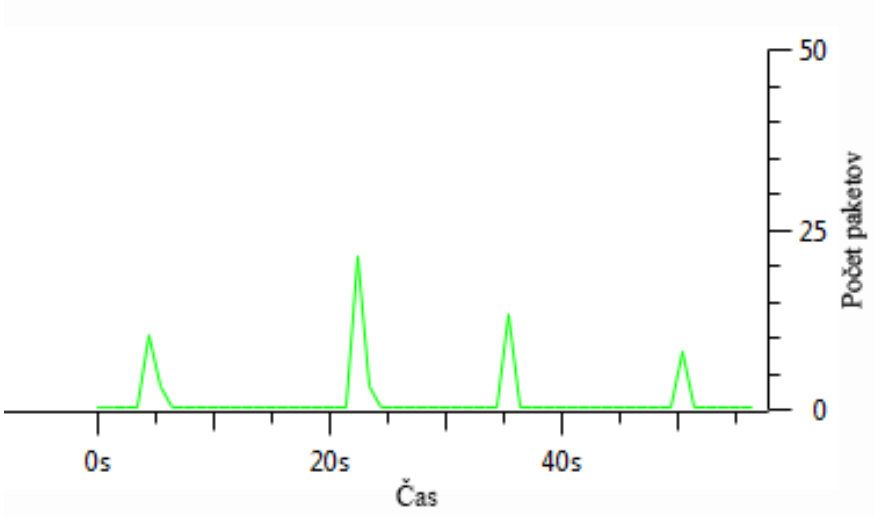
```
-- Registered SIP 'WebRTCClient2' at 10.0.1.238:10060
```

Následne som vytvoril audio hovor zo sipML5 klienta „WebRTCClient1“ na klienta sipML5 klienta „WebRTCClient2“.

```
  == Using SIP VIDEO CoS mark 6
  == Using SIP RTP CoS mark 5
    -- Executing [200@webrtc:1] Dial("SIP/WebRTCClient1-
00000004", "SIP/WebRTCClient2") in new stack
  == Using SIP VIDEO CoS mark 6
  == Using SIP RTP CoS mark 5
    -- Called SIP/WebRTCClient2
    -- SIP/WebRTCClient2-00000005 is ringing
    -- SIP/WebRTCClient2-00000005 answered SIP/WebRTCClient1-
00000004
    -- Remotely bridging SIP/WebRTCClient1-00000004 and
SIP/WebRTCClient2-00000005
  == Spawn extension (webrtc, 200, 1) exited non-zero on
'SIP/WebRTCClient1-00000004'
```

Aj v tomto prípade môžeme vidieť prácu STUN-u. Bohužiaľ RTP tok neuvidíme, pretože ako bolo spomenuté už predtým toky sipML5 klientov využívajúce technológiu WebRTC nie je vidieť.

Protokol TCP po ktorom sa realizuje spojenie so softwarovou ústredňou a ktorý využívajú websockety je graficky znázornený na obrázku 5.5. Z jeho zobrazenia je vidieť, že TCP protokol zaťažuje sieť len minimálne a len v bodoch prihlásenia, vytvorenia a ukončenia spojenia sipML5 klienta so softwarovou ústredňou Asterisk.



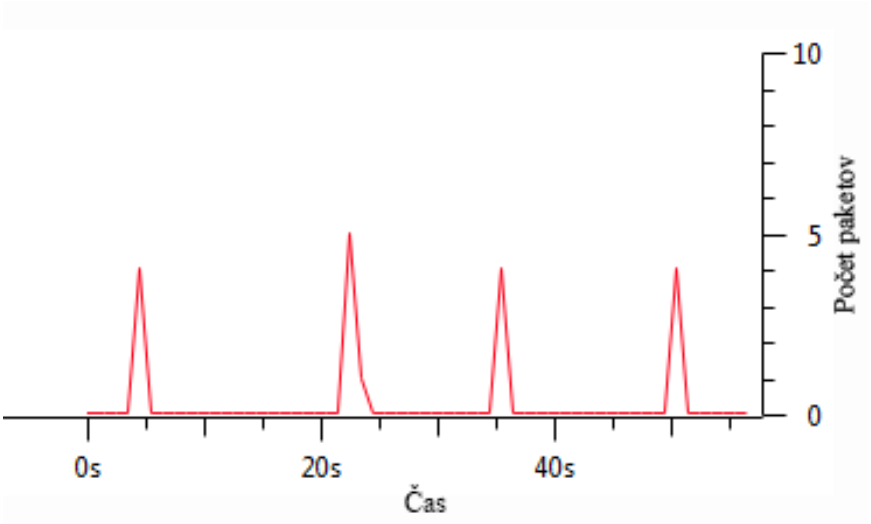
Obrázok 5.5: *Grafické zobrazenie protokolu TCP u audio hovoru medzi dvoma sipML5 klientmi*

Samotné dĺžky paketov sa pohybujú v rozmedziach ktoré sú zaznamenané v tabuľke 5.4.

Tabulka 5.4: *Dĺžka TCP paketov u audio hovoru medzi dvoma sipML5 klientmi*

Dĺžky paketov	**Množstvo**	**Percentuálny podiel**
40-79	26	40,85%
80-159	4	5,63%
160-319	9	12,68%
320-639	12	16,90%
640-1279	10	14,08%
1280-2559	7	9,86%

Na obrázku 5.6 zo spojenia môžeme vidieť zaťaženie websocketmi na sieť.



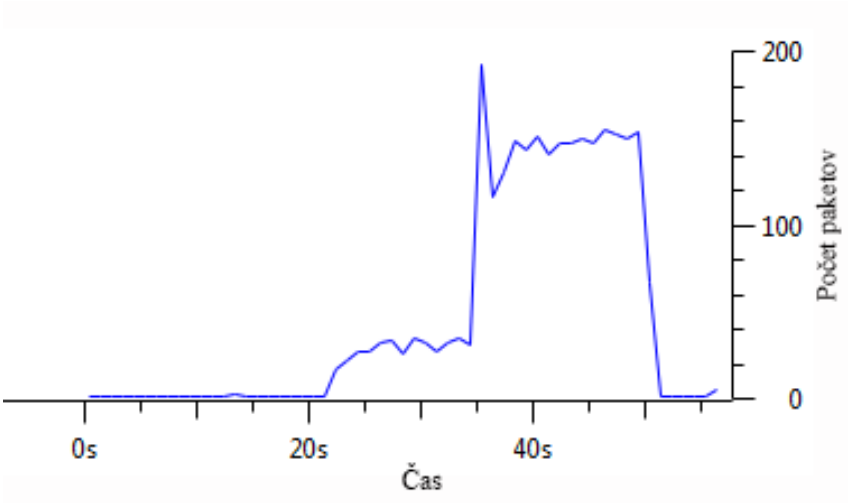
Obrázok 5.6: *Grafické znázornenie zaťaženia websocket paketami na sieť u audio hovoru medzi dvoma sipML5 klientmi*

Z obrázku 5.6 je znova možné vidieť, že websockety zaťažujú sieť len nepatrne počas celého trvania spojenia a dialógu. Samotné dĺžky paketov sa pohybujú v rozmedziach ktoré sú zaznamenané v tabuľke 5.5.

Tabulka 5.5: *Dĺžka websocket paketov u audio hovoru medzi dvoma sipML5 klientmi*

Dĺžky paketov	**Množstvo**	**Percentuálny podiel**
80-159	1	4,55%
320-639	10	45,45%
640-1279	10	45,45%
1280-2559	1	4,55%

Zaťaženie UDP paketmi na sieť je už omnoho väčšie ako je vidieť na obrázku 5.7.



Obrázok 5.7: *Grafické znázornenie zaťaženia UDP paketami na sieť u audio hovoru medzi dvoma sipML5 klientmi*

Je vidieť, že zaťaženie zo strany UDP paketov je už znateľné, pričom dĺžky paketov sa pohybujú v rozmedziach ktoré sú zobrazené v tabuľke 5.6.

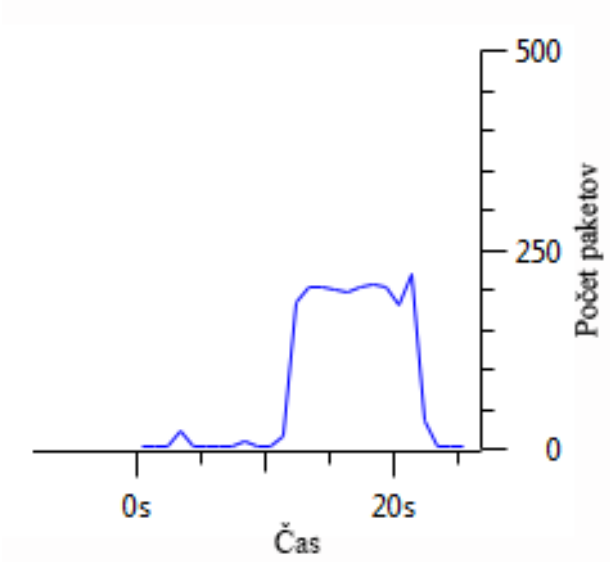
Tabulka 5.6: *Dĺžka UDP paketov u audio hovoru medzi dvoma sipML5 klientmi*

Dĺžky paketov	**Množstvo**	**Percentuálny podiel**
40-79	1	0,04%
80-159	245	9,23%
160-319	1873	70,55%
320-639	5	0,19%
640-1279	531	20,00%

5.1.3 Analýza audio hovoru medzi dvoma SIP klientmi

Audio hovor prebiehal medzi SIPClient1 klientom a SIPClient2 klientom. Celé spojenie je vidieť v paketovom analyzátore Wireshark. Taktiež bolo vidieť celý RTP. Spojenie prebiehalo podobne ako je spracované v literatúre [19]. Oproti sipML5 spojeniu už sú využívané „len“ UDP pakety. Bol použitý audio kódek ulaw g.711.

Zaťaženie UDP paketmi na sieť je vidieť na obrázku 5.8.



Obrázok 5.8: *Grafické znázornenie zaťaženia UDP paketmi na sieť u audio hovoru medzi dvoma SIP klientmi*

Jednotlivé dĺžky UDP paketov v závislosti na ich percentuálnom podieli v sieti sa nachádzajú v tabuľke 5.7.

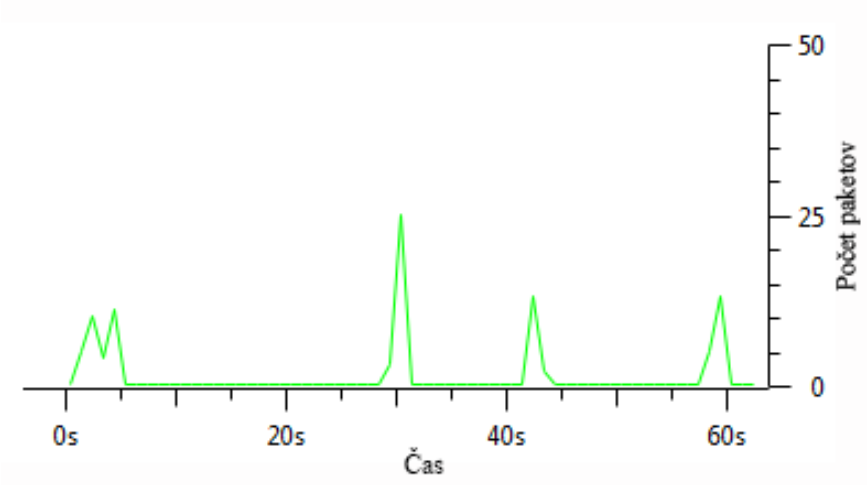
Tabulka 5.7: *Dĺžka UDP paketov u audio hovoru medzi dvoma SIP klientmi*

Dĺžky paketov	Množstvo	Percentuálny podiel
40-79	1	0,05%
80-159	5	0,24%
160-319	2007	97,19%
320-639	28	1,36%
640-1279	24	1,16%

5.1.4 Analýza video hovoru medzi dvoma sipML5 klientmi

SipML5 ponúka aj možnosť využiť video hovor. Realizácia prebiehala medzi dvoma sipML5 klientmi „WebRTCClient1“ a „WebRTCClient2“. Spojenie funguje bez problémov a na oboch stranách je vidieť video obraz. Viz. príloha C. V tomto spojení bol použitý kódek VP8.

Správy mali rovnaký charakter ako správy pri audio hovore medzi dvoma sipML5 klientmi. Tak ako aj predtým, tak aj teraz bude analýza prebiehať rovnako za pomoci programu Wireshark. Začnem analýzou TCP. Protokol TCP je zobrazený na obrázku 5.9.



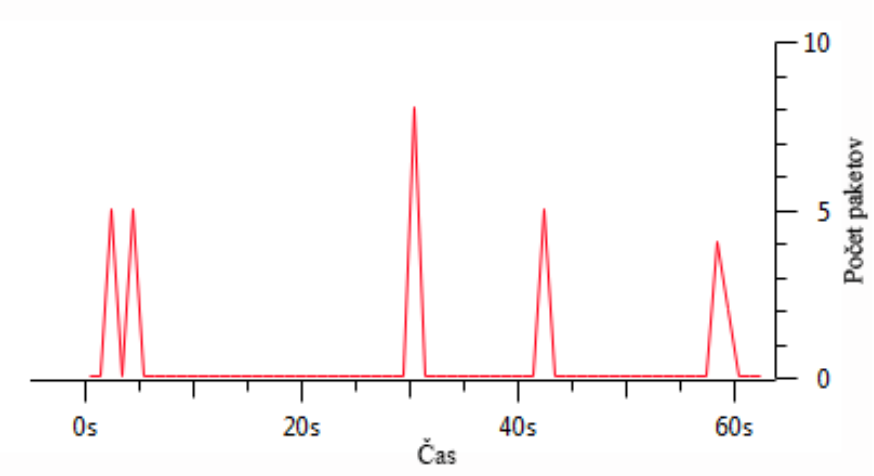
Obrázok 5.9: *Grafické znázornenie zaťaženia TCP paketami na sieť u video hovoru medzi dvoma sipML5 klientmi*

Samotné dĺžky paketov sa pohybujú v rozmedziach ktoré sú zaznamenané v tabuľke 5.8.

Tabulka 5.8: *Dĺžky TCP paketov pri video hovore medzi dvoma sipML5 klientmi*

Dĺžky paketov	Množstvo	Percentuálny podiel
40-79	38	41,76%
80-159	4	4,40%
160-319	12	13,19%
320-639	15	16,48%
640-1279	13	14,29%
1280-2559	9	9,89%

Je vidieť, že zaťaženie TCP protokolom na sieť sa moc nelíši od zaťaženia pri audio hovore. Nasleduje analýza spojenia websocketmi. Grafické zaťaženie siete je znázornené na obrázku 5.10.



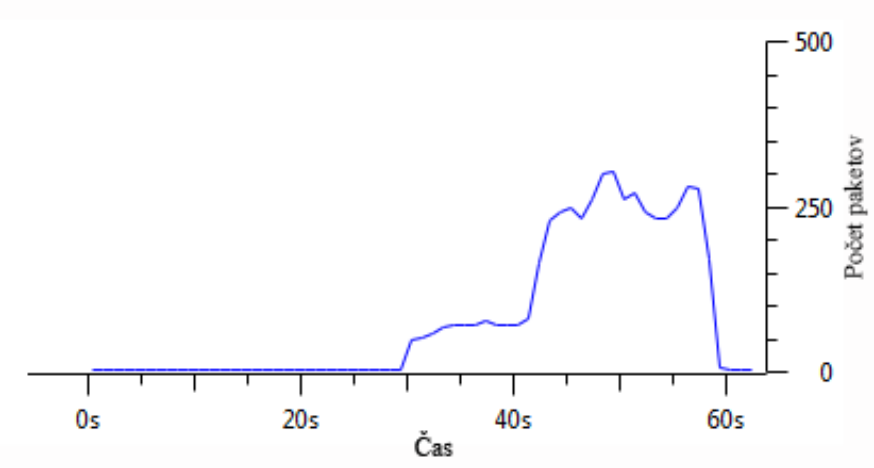
Obrázok 5.10: *Grafické znázornenie zaťaženia websocket paketmi na sieť u video hovoru medzi dvoma sipML5 klientmi*

Samotné dĺžky paketov sa pohybujú v rozmedziach ktoré sú zaznamenané v tabuľke 5.9.

Tabulka 5.9: *Dĺžky websocket paketov pri video hovore medzi dvoma sipML5 klientmi*

Dĺžky paketov	**Množstvo**	**Percentuálny podiel**
80-159	1	3,45%
160-319	2	6,90%
320-639	10	34,48%
640-1279	13	44,83%
1280-2559	3	10,34%

Ako posledná analýza je analýza UDP protokolu, ktorá je graficky znázornená na obrázku 5.11.



Obrázok 5.11: *Grafické znázornenie zaťaženia UDP paketmi na sieť u video hovoru medzi dvoma sipML5 klientmi*

Taktiež rozbor samotných dĺžok paketov sa pohybuje v rozmedziach ktoré sú zaznamenané v tabuľke 5.10.

Tabulka 5.10: *Dĺžky UDP paketov pri video hovore medzi dvoma sipML5 klientmi*

Dĺžky paketov	**Množstvo**	**Percentuálny podiel**
40-79	7	0,14%
80-159	584	11,67%
160-319	3280	65,56%
320-639	20	0,40%
640-1279	1112	22,23%

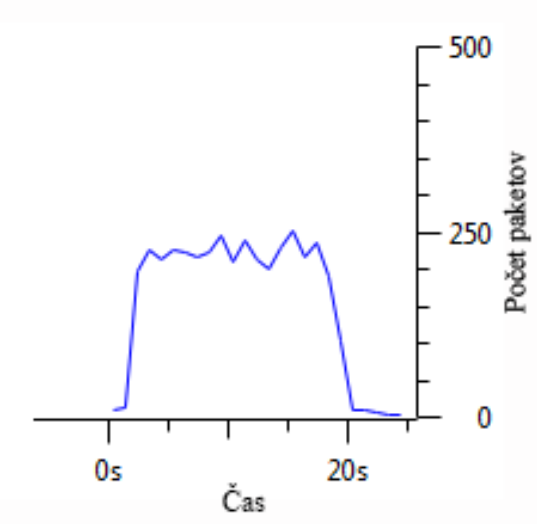
Ako by sa dalo predpokladať, tak zaťaženie UDP protokolom je omnoho väčšie keďže sa jedná o prenos videa.

Pre porovnanie vytvorím spojenie tiež medzi dvoma SIP klientmi, ktoré budú využívať nie VP8 kódek ale h.263 Plus kódek.

5.1.5 Analýza video hovoru medzi dvoma SIP klientmi

V tomto spojení som využíval kódek h.263 Plus. Celé spojenie je možné ľahko odchytiť v paketovom analyzátore Wireshark. Odchytený bol celý RTP tok a taktiež kódek h.263 Plus. Odchytený h.263 Plus paket je možné vidieť v prílohe D.

Je jasné, že toto spojenie využíva len UDP protokol. Jeho grafické znázornenie zaťaženia na sieť je na obrázku 5.12.



Obrázok 5.12: *Grafické znázornenie zaťaženia UDP paketami na sieť u video hovoru medzi dvoma SIP klientmi*

Samotné dĺžky jednotlivých paketov sa pohybujú v rozmedziach, ktoré sú zaznamenané v tabuľke 5.11.

Tabulka 5.11: *Dĺžky UDP paketov pri video hovore medzi dvoma SIP klientmi*

Dĺžky paketov	Množstvo	Percentuálny podiel
40-79	2	0,05%
80-159	44	1,14%
160-319	2968	76,71%
320-639	73	1,89%
640-1279	350	11,17%
1280-2559	432	11,17%

Počas jednotlivých analýz meraní sme si mohli všimnúť pomalší nábeh spojenia u sipML5 klientov hlavne na začiatku pri inicializácii spojenia. Pre objasnenie to je spôsobené STUN.

Dalo by sa povedať, že nevýhodou SIP (Session Initiation Protocol) oproti WebRTC je možnosť odchytiť celú konverzáciu. I keď v súčasnosti nie je možné dekódovať už odchytený hovor u SIP pri SRTP a ZRTP.

5.2 Porovnanie časov

V neposlednom rade som sa rozhodol ešte porovnať časy od odoslania správy INVITE (volajúceho) inicializátora spojenia až po odoslanie správy Ringing zo strany prijímajúceho (volaného) klienta. Na začiatku som si neplánoval zvoliť práve tieto časy ale žiaľ v paketovom analyzátore Wireshark som nemohol porovnať iné kvôli tomu, že som nemohol vidieť ich časy (nevidel som paket ale videl som len obsah správy. Menu „*Analyze – Follow TCP stream"*). Z tohto dôvodu som si zvolil nasledujúce. Jednotlivé časy sa nachádzajú v tabuľke 5.12.

Tabulka 5.12: *Uplynuté časy medzi správami INVITE a Ringing v jednotlivých typoch spojenia*

Typ spojenia	Čas odoslania správy INVITE [s]	Čas odoslania správy Ringing [s]	Rozdiel časov [s]
sipML5 => sipML5 Audio	22,7223	23,0029	0,2806
SIP klient => SIP klient Audio	8,5457	8,8886	0,3429
sipML5 => sipML5 Video	30,0826	30,5267	0,4441
SIP klient => SIP klient Video	0,0326	0,4620	0,4294

Z tabuľky 5.12 je vidieť, že rozdiely časov sa od seba príliš nelíšia. Povedal by som, že cieľom technológie nebolo zrýchliť prenos správ ale hlavne uľahčiť prístup koncových účastníkov na vysokej bezpečnostnej úrovni.

6 Zhodnotenie

Technológia WebRTC je sľubnou technológiou, ktorá sa v budúcnosti bude len rozvíjať. V spolupráci s projektom sipML5 predpokladám budú tvoriť výborné riešenie pre komunikáciu cez webový prehliadač. Veľkou výhodou je, že je zatiaľ založená na voľne šíriteľných zdrojoch.

Projekt sipML5 v spolupráci so softvérovou ústredňou Asterisk tvorí výbornú „dvojičku“ a komunikácia medzi klientmi a serverom je na vysokej bezpečnostnej úrovni.

Povedal by som, že cieľom technológie nebolo zrýchliť alebo pozmeniť prenos správ ale hlavne uľahčiť prístup koncových účastníkov na vysokej bezpečnostnej úrovni. Dokazuje to, to že pri komunikácii za pomoci technológie WebRTC nie je vidieť žiaden RTP tok. Vysvetlením je to, že kvôli možnosti využívať webový prehliadač ako VoIP komunikátor sa javí ako jednoduchá možnosť podvrhnutia a odposluchu. A tak v paketovom analyzátore Wireshark neodchytíme žiadny tok RTP. V tomto vidím obrovskú výhodu technológie WebRTC oproti predošlej.

Taktiež musím spomenúť, že poteší jednotnosť a jednoduchosť prostredia v ktorom sa koncový užívateľ pohybuje.

Na druhú stranu by som ale rád spomenul, že projektu sipML5 v spolupráci s Asteriskom chýba vo veľkej miere dokumentácia ako všetko spustiť tak, aby jednotlivé časti pracovali správne. Pri kompilácii som sa osobne stretol s veľkým množstvom problémov a práve to môže správcu takejto technológie odvrátiť od jej implementácie. Taktiež sa tu naskytol problém s kompatibilitou či už s verziami Asterisku alebo kódekmi.

Záver

Cieľom diplomovej práce bolo popísať pokročilú technológiu, ktorú dnešné moderné telefónne ústredne môžu zaistiť a jej implementácia pre softvérovú ústredňu Asterisk. Výstupom je samotné nastavenie, ktoré zaisťuje plnú funkcionalitu a interoperabilitu použitých komponent s Asteriskom. Integráciou tejto technológie do už používaných informačných systémov je možné docieliť rýchlejšej, kvalitnejšej a hlavne bezpečnejšej či už internej komunikácie tak aj v externej komunikácii spoločnosti napríklad so svojimi klientmi.

Dôležitou súčasťou tejto novej technológie musí byť aj kompatibilita so svojím predchodcom hojne používaným SIP protokolom. Keďže WebRTC predstavuje len samotnú technológiu tak z dôvodu praktickej časti bola táto technológia využívaná zatiaľ otvoreným a voľne šíriteľným projektom sipML5 od spoločnosti Doubango. Žiaľ implementácia sipML5 projektu priamo do softvérovej ústredne nemá stále nejaký oficiálny spôsob kompilácie a spustenia. S týmto vzrastajú aj problémy s kompatibilitou v rôznych verziách Asterisku alebo verziách webového prehliadača Google Chrome. Čiže vo veľkej miere chýba štandardizácia celého projektu sipML5 v spolupráci s Asteriskom.

Technológia WebRTC je obohatená aj o nové kódeky, tri audio a jeden video kódek. No naskytuje sa tu problém s kompatibilitou. To ale neznamená, že WebRTC nepodporuje už stávajúce kódeky ako napríklad ulaw, alaw.

Nová technológia prichádza aj s novým typom paketov použitých či už k registrácii pri ktorej dochádza k prepnutiu protokolu alebo aj nadviazaniu a ukončeniu spojenia. Tento nový typ sa nazýva websockety. Čo sa týka ich obsahu tak sa od obsahu paketov použitých pri SIP protokole vôbec nelíšia. Websockety som spracoval a ich obsah je možné vidieť v prílohe.

Po detailnom pohľade na technológiu WebRTC v spolupráci so softvérovou ústredňou Asterisk a sipML5 sa vývojári snažili dbať hlavne na bezpečnosť, ktorá je oproti predchádzajúcemu protokolu omnoho výraznejšia. Spojenia sa kvalitatívne moc nelíšia a ak tak, tak len minimálne. V neposlednom rade treba spomenúť nové kódeky. Tie bohužiaľ ešte stále nepracujú ako by si samotní tvorcovia predstavovali. No vyriešenie kompatibility by nemalo byť veľkým problémom. Najväčšou nevýhodou je štandardizácia celého projektu sipML5 so softvérovou ústredňou Asterisk. Musím povedať, že je možné sa tu stretnúť s množstvom problémov pokiaľ sa rozhodneme použiť čo i len inú verziu Asterisku. Dátum štandardizácie uvádza komunita Doubango Telecom na rok 2018.

Projekt sipML5 sa zameriava aj na koncových účastníkov. Na koncových účastníkov viditeľné rozhranie od sipML5 pôsobí veľmi jednoducho. Na toto je dôležité poukázať pretože práve koncový účastník sa bude s týmto rozhraním stretávať najčastejšie.

Pokiaľ sa jedná o porovnanie konkurentov projektu sipML5, tak jej najväčším je program Skype od spoločnosti Microsoft. Pravdou je ale to, že Skype je ponúkaný treťou stranou cez ktorú celý dátový tok prechádza či už audio alebo video. No z pohľadu projektu

sipML5 žiadna ďalšia tretia strana cez ktorú dátový tok ide neexistuje. Celá komunikácia prechádza cez softvérovú ústredňu Asterisk, ktorú si môžeme sami spravovať.

Použitá literatúra

[1] AGORA LABS. *VoIP multimedia* [online]. 2008 [cit. 2014-04-23]. Dostupné z: http://www.agoralabs.com/speech-codec/g722-codec.htm

[2] *Asterisk and sipml5 interoperability* [online]. 2013 [cit. 2014-03-18]. Dostupné z: http://linux.autostatic.com/asterisk-and-sipml5-interoperability

[3] BRESCIANI, Riccardo. *The ZRTP Protocol Security Considerations* [online]. 2007 [cit. 2014-03-10]. Dostupné z: http://www.lsv.ens-cachan.fr/Publis/RAPPORTS_LSV/PDF/rr-lsv-2007-20.pdf

[4] DIGIUM, Inc. *Asterisk – New in 12*. [online]. 2013 [cit. 2014-02-11]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/New+in+12

[5] DIGIUM, Inc. *Asterisk.* [online]. 2013 [cit. 2013-07-07]. Dostupné z: http://www.asterisk.org/

[6] DIGIUM, Inc. *Asterisk wiki.* [online]. 2013 [cit. 2013-07-07]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/Home

[7] DIGIUM, Inc. *Asterisk project: System Requirements* [online]. [cit. 2014-03-26]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/System+Requirements

[8] COLP, Joshua. DIGIUM, Inc. *WebRTC* [online]. 2012 [cit. 2014-04-20]. Dostupné z: https://wiki.asterisk.org/wiki/display/~jcolp/WebRTC

[9] IETF RTP: A Transport Protocol for Real-Time Applications. 75s. Dostupné z: https://www.ietf.org/rfc/rfc1889.txt

[10] IETF. *The Secure Real-time Transport Protocol (SRTP): RFC3711*. 2004, 56 s. Dostupné z: http://tools.ietf.org/html/rfc3711

[11] IETF. *ZRTP: Media Path Key Agreement for Unicast Secure RTP: RFC6189*. 2011, 111 s. Dostupné z: http://tools.ietf.org/html/rfc6189

[12] IETF. *The WebSocket Protocol as a Transport for the Session Initiation Protocol (SIP)*. 16.04.2012, 19 s. Dostupné z: http://tools.ietf.org/html/draft-ibc-sipcore-sip-websocket-02

[13] INTERNET ENGINEERING TASK FORCE (IETF). *The WebSocket Protocol*. 2011, 71 s. Dostupné z: https://tools.ietf.org/html/rfc6455

[14] MIKULEC, Martin. *Pokročilé služby v podnikové komunikaci*. Ostrava, 2011. Diplomová práca. VŠB – Technická univerzita Ostrava. Vedoucí práce doc. Ing. Miroslav Vozňák, Ph.D.

[15] *RTP statistics: How jitter is calculated* [online]. 2009 [cit. 2014-03-29]. Dostupné z: http://wiki.wireshark.org/RTP_statistics

[16] *RTC Hardware Coding Requirements* [online]. 2010 [cit. 2014-04-15]. Dostupné z: http://www.webmproject.org/hardware/rtc-coding-requirements/

[17] *Sipml5: The world's first HTML5 SIP client (WebRTC)* [online]. 2013, 16. 3. 2013 [cit. 2014-03-18]. Dostupné z: https://code.google.com/p/sipml5/wiki/Asterisk

[18] *Tap-rtp-common.c: RTP stream handler functions used by tshark and wireshark*. 2003. Dostupné z: https://cs.fit.edu/code/projects/cse2410f13team7/repository/entry/wireshark/ui/tap-rtp-common.c#L627

[19] VOZŇÁK, Miroslav. *Voice over IP*. Ostrava, 2012. Skripta. VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA

[20] *WebRTC* [online]. 2011 [cit. 2014-04-15]. Dostupné z: http://www.webrtc.org/reference/architecture#TOC-iSAC-iLBC-Opus

[21] WIESNER, David. *VIDEOKONFERENCE V HTML 5*. Brno, 2013. BAKALÁRSKÁ PRÁCE. VYSOKÉ UCENÍ TECHNICKÉ V BRNE. Vedúci práce Ing. PETR CÍKA, Ph.D.

[22] *Webrtc2sip* [online]. 2012, 27.4.2014 [cit. 2014-05-03]. Dostupné z: https://code.google.com/p/webrtc2sip/wiki/Building_Source_v2_0

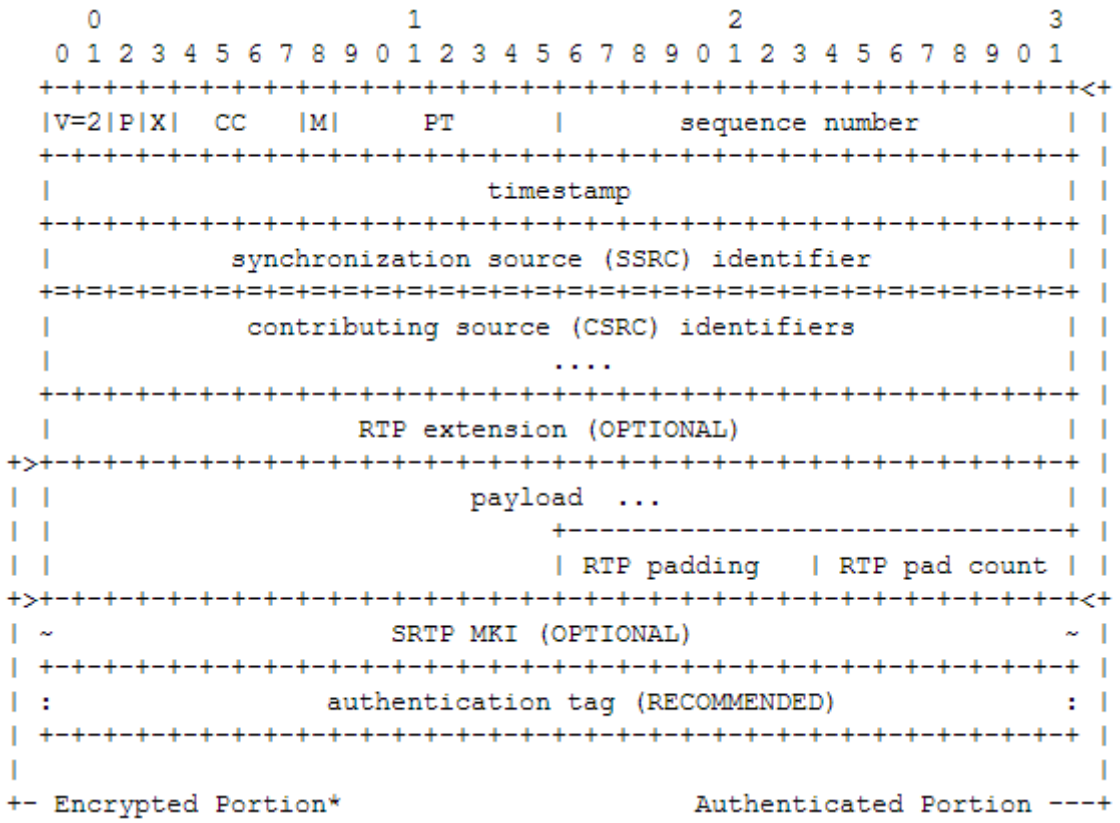
[23] GOOGLE. *Prehliadač Chrome* [online]. [cit. 2014-05-05]. Dostupné z: https://www.google.com/intl/sk/chrome/browser/

[24] COUNTERPATH. *X-lite* [online]. 2003 [cit. 2014-05-05]. Dostupné z: https://www.counterpath.com/x-lite.html

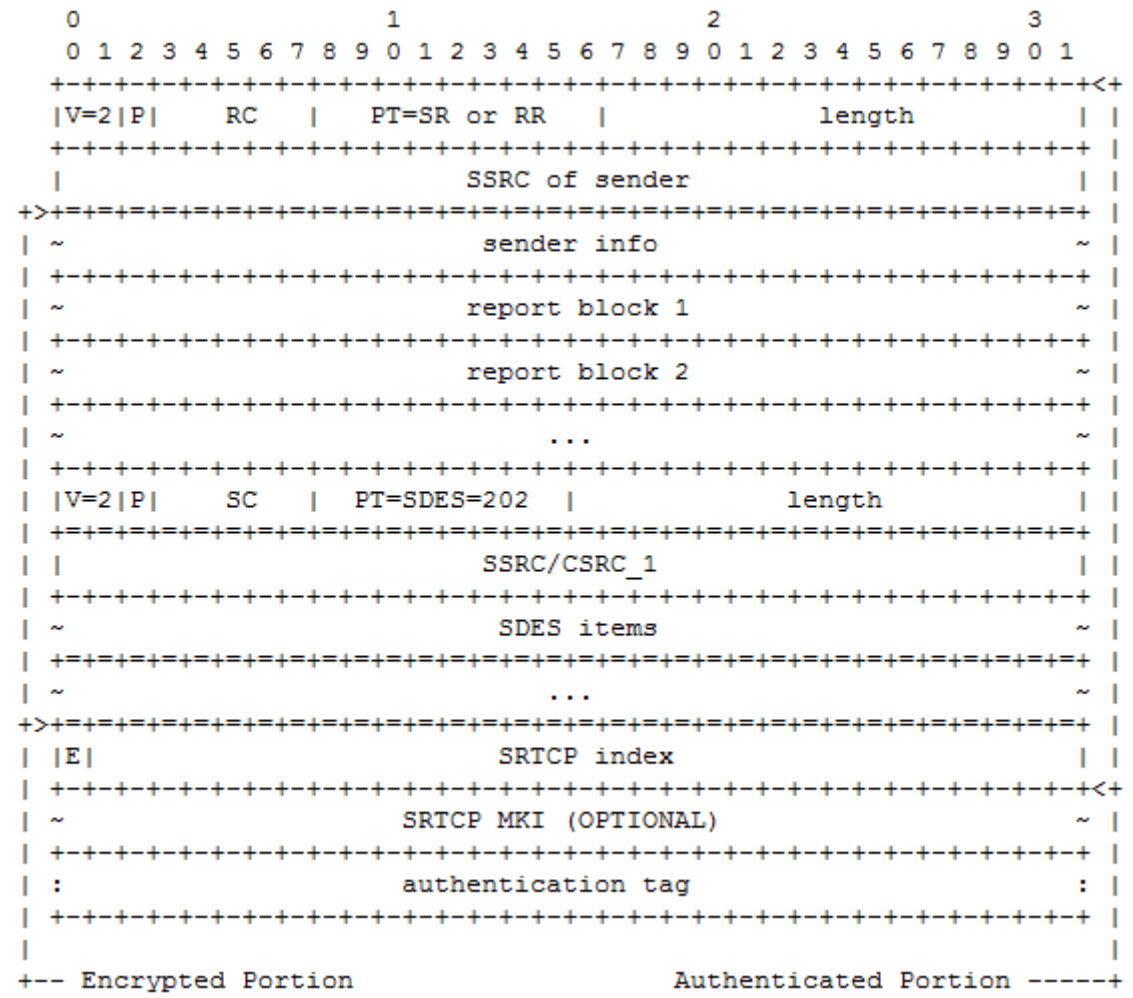
Zoznam príloh

Príloha A: *Formáty rámcov*

Obrázok A.1: *Formát rámca SRTP*



Obrázok A.2: *Formát rámca SRTCP*



Príloha B: *Správy spojenia*

Výpis správy spojenia B.1: *Správa INVITE*

```
INVITE sip:300@10.0.1.238 SIP/2.0

Via: SIP/2.0/WS
df7jal23ls0d.invalid;branch=z9hG4bKxMizyhVL44fJF8EEExFN5ACTpBTeG
50F;rport

From:
"WebRTCClient1"<sip:WebRTCClient1@10.0.1.238>;tag=1B24UXCS6CnOZg
pne9PW

To: <sip:300@10.0.1.238>
```

```
Contact:
"WebRTCClient1"<sip:WebRTCClient1@df7jal23ls0d.invalid;rtcweb-
breaker=yes;click2call=no;transport=ws>;impi=WebRTCClient1;ha1=d
768724d783a72b6ad5fc1725c21929f;+g.oma.sip-
im;+sip.ice;language="en,fr"
Call-ID: 57b28926-f4eb-7387-5ca4-4be7345477d7
CSeq: 7202 INVITE
Content-Type: application/sdp
Content-Length: 2574
Max-Forwards: 70
User-Agent: IM-client/OMA1.0 sipML5-v1.2014.04.18
Organization: Doubango Telecom

v=0
o=- 7642242693079016000 2 IN IP4 127.0.0.1
s=Doubango Telecom - chrome
t=0 0
a=group:BUNDLE audio
a=msid-semantic: WMS b4yshPU8DN7N0uFwc64QFLZ1MvKxgOPUBFCi
m=audio 60916 RTP/SAVPF 111 103 104 0 8 106 105 13 126
c=IN IP4 158.196.42.141
a=rtcp:60916 IN IP4 158.196.42.141
a=candidate:3997094279 1 udp 2122063615 158.196.42.141 60916 typ
host generation 0
a=candidate:3997094279 2 udp 2122063615 158.196.42.141 60916 typ
host generation 0
a=candidate:4060033307 1 udp 2121998079 192.168.25.1 60917 typ
host generation 0
a=candidate:4060033307 2 udp 2121998079 192.168.25.1 60917 typ
host generation 0
a=candidate:1517646798 1 udp 2121932543 10.2.3.18 60918 typ host
generation 0
a=candidate:1517646798 2 udp 2121932543 10.2.3.18 60918 typ host
generation 0
```

```
a=candidate:3200396804 1 udp 2121867007 192.168.72.1 60919 typ
host generation 0

a=candidate:3200396804 2 udp 2121867007 192.168.72.1 60919 typ
host generation 0

a=candidate:2696752503 1 tcp 1518083839 158.196.42.141 0 typ
host generation 0

a=candidate:2696752503 2 tcp 1518083839 158.196.42.141 0 typ
host generation 0

a=candidate:3212627435 1 tcp 1518018303 192.168.25.1 0 typ host
generation 0

a=candidate:3212627435 2 tcp 1518018303 192.168.25.1 0 typ host
generation 0

a=candidate:351702846 1 tcp 1517952767 10.2.3.18 0 typ host
generation 0

a=candidate:351702846 2 tcp 1517952767 10.2.3.18 0 typ host
generation 0

a=candidate:4030845684 1 tcp 1517887231 192.168.72.1 0 typ host
generation 0

a=candidate:4030845684 2 tcp 1517887231 192.168.72.1 0 typ host
generation 0

a=ice-ufrag:Z17bSD4lRDZ+fmb1

a=ice-pwd:ORjWSLNeTsRHN2dLuC+lFs8/

a=ice-options:google-ice

a=fingerprint:sha-256
0E:8E:E5:6E:F5:4B:EF:97:32:99:A1:66:3B:74:A5:48:B3:48:11:E1:44:0
5:77:BC:D3:0B:FA:B6:C3:48:D5:07

a=setup:actpass

a=mid:audio

a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level

a=sendrecv

a=rtcp-mux

a=crypto:0 AES_CM_128_HMAC_SHA1_32
inline:37uhlRV14OWmCev58k7n8JKItG9GoBTgoeHAg6w8

a=crypto:1 AES_CM_128_HMAC_SHA1_80
inline:n2DEjkLUQE0qF+phnYF2T2ehzAhsWlKN2P4rsEd8
```

```
a=rtpmap:111 opus/48000/2
a=fmtp:111 minptime=10
a=rtpmap:103 ISAC/16000
a=rtpmap:104 ISAC/32000
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:106 CN/32000
a=rtpmap:105 CN/16000
a=rtpmap:13 CN/8000
a=rtpmap:126 telephone-event/8000
a=maxptime:60
a=ssrc:2002878253 cname:Gch8OdXdZoUhRHDg
a=ssrc:2002878253 msid:b4yshPU8DN7N0uFwc64QFLZ1MvKxgOPUBFCi
8729ca8c-9416-40e6-8dff-3a2d6e76b73b
a=ssrc:2002878253 mslabel:b4yshPU8DN7N0uFwc64QFLZ1MvKxgOPUBFCi
a=ssrc:2002878253 label:8729ca8c-9416-40e6-8dff-3a2d6e76b73b
```

Výpis správy spojenia B.2: *Správa TRYING*

```
SIP/2.0 100 Trying (sent from the Transaction Layer)
Via: SIP/2.0/WS
df7jal23ls0d.invalid;rport;branch=z9hG4bKxMizyhVL44fJF8EEExFN5AC
TpBTeG50F
From:
"WebRTCClient1"<sip:WebRTCClient1@10.0.1.238>;tag=1B24UXCS6CnOZg
pne9PW
To: <sip:300@10.0.1.238>
Call-ID: 57b28926-f4eb-7387-5ca4-4be7345477d7
CSeq: 7202 INVITE
Content-Length: 0
```

Výpis správy spojenia B.3: *Správa RINGING*

```
SIP/2.0 180 Ringing

Via: SIP/2.0/WS
df7jal23ls0d.invalid;rport;branch=z9hG4bKxMizyhVL44fJF8EEExFN5AC
TpBTeG50F

From:
"WebRTCClient1"<sip:WebRTCClient1@10.0.1.238>;tag=1B24UXCS6CnOZg
pne9PW

To: <sip:300@10.0.1.238>;tag=337767072

Contact: <sip:300@10.0.1.238:10060;transport=ws;ws-src-
ip=10.2.3.18;ws-src-port=2732;ws-src-proto=ws>

Call-ID: 57b28926-f4eb-7387-5ca4-4be7345477d7

CSeq: 7202 INVITE

Content-Length: 0

Allow: ACK, BYE, CANCEL, INVITE, MESSAGE, NOTIFY, OPTIONS,
PRACK, REFER, UPDATE
```

Výpis správy spojenia B.4: *Správa 200 OK*

```
SIP/2.0 200 OK

Via: SIP/2.0/WS
df7jal23ls0d.invalid;rport;branch=z9hG4bKxMizyhVL44fJF8EEExFN5AC
TpBTeG50F

From:
"WebRTCClient1"<sip:WebRTCClient1@10.0.1.238>;tag=1B24UXCS6CnOZg
pne9PW

To: <sip:300@10.0.1.238>;tag=337767072

Contact: <sip:300@10.0.1.238:10060;transport=ws;ws-src-
ip=10.2.3.18;ws-src-port=2732;ws-src-proto=ws>

Call-ID: 57b28926-f4eb-7387-5ca4-4be7345477d7

CSeq: 7202 INVITE

Content-Type: application/sdp

Content-Length: 805

Allow: ACK, BYE, CANCEL, INVITE, MESSAGE, NOTIFY, OPTIONS,
PRACK, REFER, UPDATE
```

```
v=0
o=doubango 1983 678901 IN IP4 10.0.1.238
s=-
c=IN IP4 10.0.1.238
t=0 0
m=audio 40842 RTP/SAVPF 0 8 126
c=IN IP4 10.0.1.238
a=ptime:20
a=minptime:1
a=maxptime:255
a=silenceSupp:off - - - -
a=rtpmap:0 PCMU/8000/1
a=rtpmap:8 PCMA/8000/1
a=rtpmap:126 telephone-event/8000/1
a=fmtp:126 0-16
a=crypto:0 AES_CM_128_HMAC_SHA1_32
inline:VSb1tfOd/o4A3U7NNdmSbkznJ9fkC1l7r3LerKgs
a=sendrecv
a=rtcp-mux
a=ssrc:2995005095 cname:f3556ba95a57a55f27173dc4b93be90f
a=ssrc:2995005095 mslabel:6994f7d1-6ce9-4fbd-acfd-84e5131ca2e2
a=ssrc:2995005095 label:doubango@audio
a=ice-ufrag:0PgXYBXjuWIk0nP
a=ice-pwd:8k0ceGIPVTkPI4WCpZmG2x
a=candidate:7f79vpFys 1 udp 2130706431 10.0.1.238 40842 typ host
a=candidate:srflx7f79 1 udp 1694498815 158.196.244.180 32413 typ
srflx raddr 10.0.1.238 rport 40842
```

Výpis správy spojenia B.5: *Správa ACK*

```
ACK sip:300@10.0.1.238:10060;transport=ws;ws-src-
ip=10.2.3.18;ws-src-port=2732;ws-src-proto=ws SIP/2.0

Via: SIP/2.0/WS
df7jal23ls0d.invalid;branch=z9hG4bKx5TeivvWMBylIrLv30o7;rport

From:
"WebRTCClient1"<sip:WebRTCClient1@10.0.1.238>;tag=1B24UXCS6CnOZg
pne9PW

To: <sip:300@10.0.1.238>;tag=337767072

Contact:
"WebRTCClient1"<sip:WebRTCClient1@df7jal23ls0d.invalid;rtcweb-
breaker=yes;click2call=no;transport=ws>;+g.oma.sip-
im;+sip.ice;language="en,fr"

Call-ID: 57b28926-f4eb-7387-5ca4-4be7345477d7

CSeq: 7202 ACK

Content-Length: 0

Max-Forwards: 70

User-Agent: IM-client/OMA1.0 sipML5-v1.2014.04.18

Organization: Doubango Telecom
```

Výpis správy spojenia B.6: *Správa BYE*

```
BYE sip:WebRTCClient1@df7jal23ls0d.invalid;rtcweb-
breaker=yes;click2call=no;transport=ws SIP/2.0

Via: SIP/2.0/WS 10.0.1.238:10060;branch=z9hG4bK-1538266804;rport

From: <sip:300@10.0.1.238>;tag=337767072

To:
"WebRTCClient1"<sip:WebRTCClient1@10.0.1.238>;tag=1B24UXCS6CnOZg
pne9PW

Call-ID: 57b28926-f4eb-7387-5ca4-4be7345477d7

CSeq: 1628636007 BYE

Content-Length: 0

Max-Forwards: 70

Route: <sip:10.2.3.18:2732;transport=ws;lr>
```

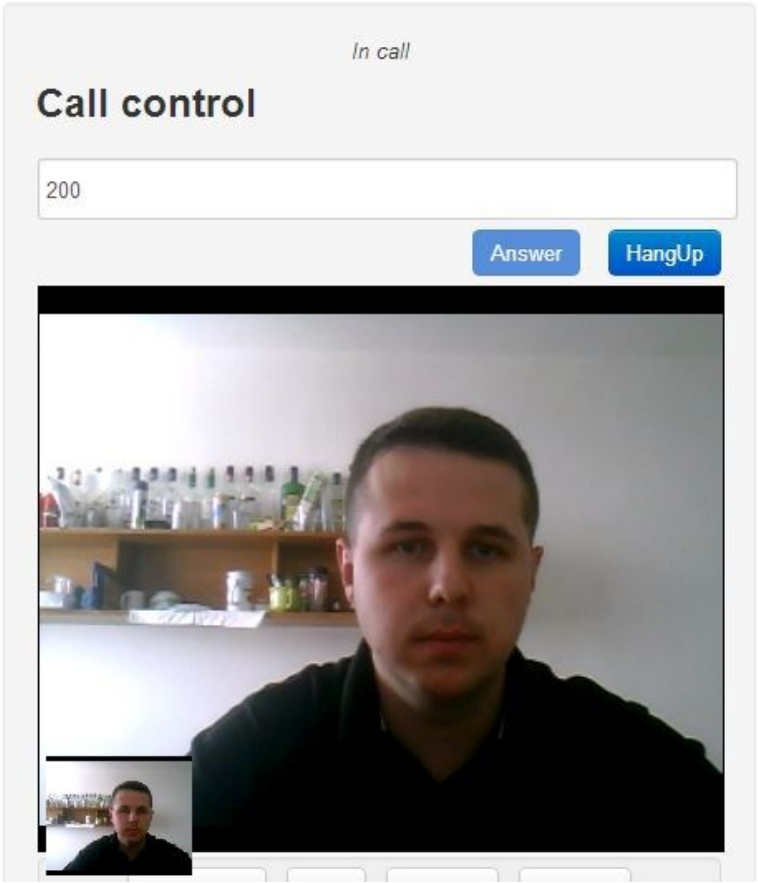
```
User-Agent: webrtc2sip Media Server 2.6.0
```

Výpis správy spojenia B.7: *Správa 200 OK*

```
SIP/2.0 200 OK
Via: SIP/2.0/WS 10.0.1.238:10060;rport=10060;branch=z9hG4bK-
1538266804
From: <sip:300@10.0.1.238>;tag=337767072
To:
"WebRTCClient1"<sip:WebRTCClient1@10.0.1.238>;tag=1B24UXCS6CnOZg
pne9PW
Contact: <sip:WebRTCClient1@df7jal23ls0d.invalid;transport=ws>
Call-ID: 57b28926-f4eb-7387-5ca4-4be7345477d7
CSeq: 1628636007 BYE
Content-Length: 0
```

Príloha C: *Ukážka video hovoru cez sipML5 s technológiou WebRTC*

Obrázok C.1: *Video hovor na jednej strane sipML5 klienta*



Príloha D: *Odchytený paket h.263 Plus*

Obrázok D.1: *Paketu h.263 Plus*

```
H.263 RFC4629 payload
    1000 00.. = H.263 Picture start Code: 0x00000020
    .... ..00  0000 01.. = H.263 Temporal Reference: 1
    0... .... = H.263 Split screen indicator: Off
    .0.. .... = H.263 Document camera indicator: Off
    ..0. .... = H.263 Full Picture Freeze Release: Off
    ...0 11.. = H.263 Source Format: CIF 352x288 (0x03)
    .... ..1. = H.263 Picture Coding Type: INTER (P-picture)
    .... ...0 = H.263 Optional Unrestricted Motion Vector mode: Off
    0... .... = H.263 Optional Syntax-based Arithmetic Coding mode: Off
    .0.. .... = H.263 Optional Advanced Prediction mode: Off
    ..0. .... = H.263 Optional PB-frames mode: Normal I- or P-picture
    ...0 0101 = H.263 Quantizer Information (PQUANT): 5
    0... .... = H.263 Continuous Presence Multipoint and Video Multiplex (CPM): Off
    .0.. .... = H.263 Extra Insertion Information (PEI): False
```